UQAC
Université du Québec
à Chicoutimi

MAISONS INTELLIGENTES : NOUVELLES APPLICATIONS DES *WEARABLE DEVICES* DANS UNE ARCHITECTURE DISTRIBUÉE GRÂCE À UN OUTIL D'APPRENTISSAGE MACHINE MODULAIRE

PAR FLORENTIN THULLIER

THÈSE PRÉSENTÉE À L'UNIVERSITÉ DU QUÉBEC À CHICOUTIMI COMME EXIGENCE PARTIELLE EN VUE DE L'OBTENTION DU GRADE DE PHILOSOPHIÆ DOCTOR (PH.D.) EN SCIENCES ET TECHNOLOGIES DE L'INFORMATION

QUÉBEC, CANADA

RÉSUMÉ

Le vieillissement que connaît la population, majoritairement, celle des pays développés, entraîne inévitablement un plus grand nombre de personnes dont l'autonomie se retrouve fortement diminuée, voire complètement perdue. Ce déclin d'autonomie peut être corrélé aux maladies neurodégénératives comme la maladie d'Alzheimer, mais également aux différents handicaps. Selon la gravité de la perte d'autonomie engendrée par ces pathologies, une assistance rigoureuse demeure nécessaire, car les personnes touchées requièrent une prise en charge relativement constante. Actuellement, les acteurs de la prise en charge des malades sont majoritairement leurs proches. Néanmoins, ceux-ci doivent alors assumer les conséquences aussi bien sur le plan personnel et émotionnel qu'au niveau social et financier.

Les récents progrès en matière de technologies de l'information et de microélectronique ont permis de faire émerger de nouveaux concepts comme celui de l'Intelligence Ambiante (IAm). Ce concept peut être traduit par une couche d'abstraction où l'informatique se retrouve au service de la communication entre les objets et les personnes. En pratique, et dans une optique de court terme, l'Intelligence Ambiante est appliquée dans les habitats intelligents qui demeurent actuellement d'excellents vecteurs d'assistance pour les personnes touchées par une perte d'autonomie partielle ou totale. De plus, s'ils permettent de compenser les lourdes dépenses que représente leur prise en charge pour les systèmes de santé, ils peuvent également soulager les proches aidants vis-à-vis de la quantité de stress qu'ils éprouvent. Cependant, étant donné la diversité des conceptions qui ont été proposées pour ces habitats, il peut s'avérer parfois complexe d'y intégrer de nouvelles méthodes de reconnaissance qui exploitent les technologies les plus récentes comme les *wearable devices*. En outre, il est apparu que certains de ces habitats souffrent de problèmes de fiabilité qui pourraient mener à des situations dangereuses pour la sécurité des résidents.

Le projet de recherche présenté dans le cadre de cette thèse propose de nouvelles approches pour améliorer l'Intelligence Ambiante au sein de maisons intelligentes lorsqu'elle est réalisée avec des *wearable devices*. Afin d'atteindre cet objectif, la première solution proposée introduit une nouvelle méthode de reconnaissance basée sur l'exploitation des données inertielles générées par un *wearable device*. Dans une seconde phase, deux nouveaux systèmes permettant une meilleure intégration de ces nouvelles méthodes dans les habitats intelligents ont été proposés. Le premier concerne une architecture distribuée et moderne qui apporte fiabilité, sécurité et qui demeure facilement évolutive comparativement aux différentes implémentations existantes pour ces environnements. Enfin, le second système est un *workbench* d'apprentissage machine générique et modulaire que nous avons appelé LIARA Environment for Modular Machine Learning (LE2ML). Ainsi, cet outil représente le liant entre les nouvelles applications des *wearable device* et leur utilisation au sein des habitats intelligents.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES ABRÉVIATIONS

API	Application Programming Interface
ARFF	Attribute-Relation File Format
ASCII	American Standard Code for Information Interchange
ASI	Alimentation Sans Interruption
ATT	ATTribute Protocol
ADL	Activity of Daily Living
AMQP	Advanced Message Queuing Protocol
ANN	Artificial Neural Network
AOS	Architecture Orientée Services
BADL	Basic Activity Daily Living
BAN	Body Area Network
BLE	Bluetooth Low Energy
BNF	Backus-Naur Form
BSON	Binary JavaScript Object Notation ou Binary JSON
CLI	Command Line Interface
CoAP	Constrained Application Protocol
CORBA	Common Object Request Broker Architecture
CRF	Conditional Random Field
CSV	Comma-Separated Values
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DFT	Discrete Fourier Transform
DoF	Degrees of Freedom
DOMUS	Domotique et informatique Mobile à l'Université de Sherbrooke
DRDB	Distributed Replicated Block Device
DWPD	Discrete Wavelet Package Decomposition
DWT	Discrete Wavelet Transform
ECG	électrocardiogramme
EEG	électroencéphalogramme
EMG	électromyogramme
EADL	Enhanced Activity Daily Living
FFT	Fast Fourier Transform

FSR Force Sensitive Resistor
GATT Generic ATTribute
GAP Generic Access Profile
GPS Global Positioning System
GUI Graphical User Interface
HMM Hidden Markov Model
HTML Hypertext Markup Language
HTTP HyperText Transfer Protocol
I^2C Inter-Integrated Circuit
IA Intelligence Artificielle
IADL Instrumental Activity Daily Living
IAm Intelligence Ambiante
ID3 Iterative Dichotomiser 3
IMU Inertial Measurement Unit
I/O Input/Output
IoT Internet of Things
IPVS IP Virtual Servers
JDBC Java Database Connectivity
JSON JavaScript Object Notation
***k*-NN** *k*-Nearest Neighbors
LE2ML LIARA Environment for Modular Machine Learning
LIARA Laboratoire d'Intelligence Ambiante pour la Reconnaissance d'Activités
LNCF Light Node Communication Framework
MQTT Message Queuing Telemetry Transport
NFC Near Field Communication
NFS Network File System
NoSQL SGBD non relationnel
openHAB open Home Automation Bus
OSGi Open Services Gateway initiative
P2P Peer-to-Peer
PCA Principal Component Analysis
PNN Probabilistic Neural Network
QoS Quality of Service

RAID Redundant Array of Independent Disks
REST Representational State Transfer
RF Random Forest
RFID Radio-Frequency IDentification
SGBD Système de Gestion de Base de Données
SMTP Simple Mail Transfer Protocol
SOAP Simple Object Access Protocol
SoC System on Chip
SPI Serial Peripheral Interface
SPO_2 Saturation Pulsée en Oxygène
SSE Server-Sent Events
SSL Secure Sockets Layer
SVM Support Vector Machine
TCP Transmission Control Protocol
TLS Transport Layer Security
UDP User Datagram Protocol
UQAC Université du Québec à Chicoutimi
URI Uniform Resource Identifier
URL Uniform Resource Locator
UUID Universally Unique IDentifier
VCS Version Control System
VIP IP virtuelle
VM Machine Virtuelle
VPN Virtual Private Network
VXLAN Virtual Extensible LAN
WEKA Waikato Environment for Knowledge Analysis
XML eXtensible Markup Language
XMPP Extensible Messaging and Presence Protocol
YALE Yet Another Learning Environment
YAML Yet Another Markup Language

DÉDICACE

Je dédie cette thèse aux personnes qui m'ont supporté tout au long de ce cheminement, mes collègues, mes amis, ma famille et plus particulièrement à mes grands-parents maternels.

REMERCIEMENTS

L'aboutissement de cette thèse n'aurait pas été possible sans le soutien et la présence de nombreuses personnes. Malheureusement, il m'est impossible de toutes les remercier par écrit, mais je tiens tout de même à m'adresser à celles qui ont joué un rôle important tout au long du cheminement de ce doctorat.

En tout premier lieu, je souhaite remercier le professeur Sébastien Gaboury, mon directeur de thèse, sans qui ces lignes n'auraient probablement jamais été écrites. Il a été une source constante de motivation et d'inspiration et il a su être compréhensif et profondément humain lors des périodes plus difficiles que j'ai pu rencontrer. J'admire la rigueur et le professionnalisme avec lesquels il a su diriger mes travaux et je lui souhaite le meilleur pour la direction du laboratoire et ses directions de recherche futures.

Dans un second temps, je souhaite également adresser mes sincères remerciements au professeur Sylvain Hallé, co-directeur de cette thèse, pour son extrême bienveillance, son enthousiasme et sa bonne humeur.

Ensuite, je tenais aussi à exprimer mes remerciements au professeur Abdenour Bouzouane qui m'a inspiré pendant ses cours et dont j'apprécie la sérénité ainsi que la richesse des échanges que nous avons eus.

Bien évidemment, je n'oublierai pas de remercier mes amis. Aurélien, merci à toi pour ton soutien et ta confiance. Ambre va pouvoir compter sur son super papa. Valère, Baptiste, François, merci pour nos discussions, nos débats sans filtres, c'est comme ça qu'on fait avancer les choses. Merci à vous d'avoir fait en sorte que je puisse compter sur vous en cas de besoin, vous avez été des collègues et des amis en or. Kévin, bienvenue à ta petite Lily et merci pour ta bonne humeur et les raclettes. Enfin, un grand merci global à tous ceux (et celles) que je n'ai pas mentionnés, mais que je n'ai pas oubliés.

Pour finir, un dernier remerciement, et non des moindres, à ma famille qui, malgré la distance entre nous, a su rester une source d'inspiration, de soutien, d'amour et de fierté.

CHAPITRE I

INTRODUCTION

1.1 CONTEXTE DE LA RECHERCHE

La surpopulation est l'un des principaux maux de ce XXI^{e} siècle, auquel l'humanité doit faire face. En effet, selon un rapport des Nations Unies, la population mondiale serait de 7,3 milliards aujourd'hui, alors que 5,3 milliards d'êtres humains seulement étaient recensés en 1990. De plus, les projections réalisées pour les prochaines décennies ne s'annoncent pas encourageantes ; à savoir 8,5 milliards pour 2030, 9,7 milliards pour 2050 et 11,2 milliards pour 2100 (United Nations, 2017b). Les causes de cette inflation d'envergure sont diverses et varient selon les continents. Par exemple, les pays en développement sont ceux qui montrent les plus forts taux de fertilité, tandis que les pays développés font, quant à eux, face à un certain vieillissement de leur population. Aussi, l'espérance de vie de la population mondiale s'est accrue de manière significative, et ce, principalement en raison des avancées médicales ainsi que de l'augmentation de la production agricole. En effet, les Nations Unies recensent 901 millions de personnes dont l'âge est supérieur à 60 ans. De plus, les prévisions pour les prochaines décennies annoncent 1,4 milliard d'individus en 2030 et 2,1 milliards en 2050 (United Nations, 2015).

Dans un premier temps, le vieillissement que connaît la population, majoritairement celle des pays développés, entraîne nécessairement un plus grand nombre de personnes dont

l'autonomie se retrouve fortement diminuée, voire complètement perdue. Ce déclin d'autonomie peut être corrélé aux maladies neurodégénératives (*p. ex.* la maladie d'Alzheimer, la maladie de Parkinson, la maladie de Huntington, *etc.*), mais également aux différents handicaps (handicap physique, sensoriel ou intellectuel). Selon la gravité de la perte d'autonomie engendrée par ces pathologies, une assistance rigoureuse demeure nécessaire, car les personnes touchées requièrent une prise en charge relativement constante (Alzheimer's Association, 2018). Actuellement, les acteurs de la prise en charge des malades sont majoritairement leurs proches (Paraponaris *et al.*, 2012). Néanmoins, ceux-ci doivent alors assumer les conséquences aussi bien sur le plan personnel et émotionnel qu'au niveau social et financier (Alzheimer's Association, 2018).

Dans un second temps, la forte croissance observée depuis les années 1990 jusqu'à nos jours entraîne un exode rural important. D'après un autre rapport produit par les Nations Unies, la proportion de résidents urbains devrait atteindre 61% de la population mondiale dans les prochaines années (United Nations, 2017a). Ainsi, de plus en plus de mégapoles se créent (seulement 4 villes affichaient ce statut en 1975, pour 36 aujourd'hui), et tendent à devenir de plus en plus grosses (United Nations, 2017a). En effet, d'ici à 2025 l'Asie pourrait compter, à elle seule, pas moins de 10 villes dont la population serait supérieure à 40 millions d'habitants. De plus, cette croissance démographique mènera à une dégradation globale de l'environnement et des changements climatiques importants, car il faudra produire de plus en plus de ressources vitales, ce qui provoquera indubitablement un afflux toujours plus conséquent de pollution. De la même façon, le coût global de la vie (logements, nourriture, santé, *etc.*) augmentera, à terme,

de façon manifeste (United Nations, 2017a). Par conséquent, selon un rapport d'UN-Habitat, l'urbanisation représenterait le meilleur compromis face à l'augmentation croissante de la population, car l'activité humaine se retrouverait alors concentrée dans des surfaces limitées, ce qui réduirait l'ampleur des dommages environnementaux (UNFPA, 2007). Cependant, cela ne sera possible que si l'urbanisme actuel subit des améliorations significatives, où l'objectif est la conception de villes plus compactes, intégrées socialement, connectées, qui favorisent le développement urbain et qui sont résistantes aux changements climatiques (UNFPA, 2007).

Ainsi, les récents progrès en matière de technologies de l'information et de microélectronique ont permis de faire émerger de nouveaux concepts comme celui de l'IAm. Ce concept peut être traduit par une couche d'abstraction où l'informatique se retrouve au service de la communication entre les objets et les personnes. En pratique, il s'agit d'enrichir l'environnement dans lequel l'Homme évolue avec la technologie afin d'en extraire le contexte et apprendre son comportement pour prendre des décisions et lui porter assistance (Sadri, 2011). L'Intelligence Ambiante se retrouve notamment dans les habitats intelligents, à l'intérieur desquels différents types de capteurs et d'effecteurs ambiants et portables sont utilisés afin de récolter des données. Parmi ceux-ci, il est possible de retrouver des antennes RFID, des capteurs infrarouges et ultrasons, des contacteurs magnétiques et des accéléromètres. La production de telles données permet alors d'apporter de l'assistance et du confort au résident, mais également d'évaluer sa santé mentale et physique (Rashidi et Mihailidis, 2013; Haux *et al.*, 2016; Harris et Hunter, 2016; Johnson et Ianes, 2018).

Pour l'instant et dans une optique de court terme, les habitats intelligents demeurent d'excellents vecteurs d'assistance pour les personnes touchées par une perte d'autonomie partielle ou totale. De plus, s'ils permettent de compenser les lourdes dépenses que représente leur prise en charge pour les systèmes de santé, ils peuvent également soulager les proches aidants vis-à-vis de la quantité de stress qu'ils éprouvent. À titre d'exemple, le rapport publié par Prince *et al.* (2016) rapporte que 47 millions de personnes sont actuellement affectées par un trouble neurocognitif ce qui, selon une estimation, coûterait 818 milliards de dollars US à l'échelle mondiale. Néanmoins, les habitats intelligents pourraient tendre à devenir de plus en plus indispensables à long terme. En effet, ceux-ci pourraient avoir un rôle important dans les innovations d'urbanisme à venir, afin d'atténuer les effets engendrés par la surpopulation.

Cette thèse ne traitera, cependant, que de la problématique générale de l'assistance aux personnes en perte d'autonomie. Ainsi, les sections suivantes ont pour objectif de définir les notions clés inhérentes à celle-ci.

1.2 L'ACTIVITÉ HUMAINE

La notion d'activité humaine dans la problématique de l'assistance aux personnes affectées par une perte d'autonomie demeure fondamentale. C'est pourquoi il convient de la définir de façon appropriée. Dans le domaine de la santé, il est convenu que l'utilisation du terme activité fait directement référence au concept d'activité de la vie quotidienne (Activity of Daily Living ou ADL) introduit et présenté par Katz *et al.* (1963). Il regroupe un ensemble d'activités permettant de mesurer l'habileté pour un individu à maintenir sa santé et son

autonomie (*p. ex.* préparer à manger, s'habiller, se laver, *etc.*). Ainsi, le succès ou l'échec dans la réalisation de ces tâches constitue, pour les professionnels de la santé, une mesure fiable pour quantifier la perte d'autonomie d'un patient présentant un handicap dont celle-ci en est soit une cause, soit une conséquence. Ceci permet alors d'adapter le type et le niveau d'assistance approprié pour chacun d'entre eux (Giovannetti *et al.*, 2002). Depuis les travaux de Katz *et al.* (1963), les chercheurs Lawton et Brody (1969) ont proposé un regroupement des ADLs en deux ensembles distincts qui sont :

Les activités basiques (Basic Activity Daily Living ou BADL) regroupent l'ensemble de toutes les activités fondamentales aux besoins primaires d'une personne. Par exemple, se déplacer sans aucun outil d'assistance (béquille, canne, *etc.*), se laver, se nourrir, se coucher, se lever, *etc.* Ces activités sont généralement composées de très peu d'étapes et ne requièrent aucune planification pour être accomplies avec succès.

Les activités instrumentalisées (Instrumental Activity Daily Living ou IADL) sont, quant à elles, l'ensemble des activités qui nécessitent une certaine planification et qui permettent de statuer sur l'autonomie d'une personne en société. On y retrouve, par exemple, l'appel téléphonique, la préparation d'un repas ou encore la gestion de l'argent. Aussi, ces activités requièrent souvent beaucoup plus d'étapes dans leur exécution que les BADLs.

Finalement, ce regroupement a été étendu par Rogers *et al.* (1998) afin d'y inclure un troisième et dernier ensemble d'activités laissées pour compte jusqu'alors. Il s'agit des Enhanced Activity Daily Living (EADL) qui comprend toutes les activités qui nécessitent une

certaine adaptation ou un apprentissage pour être menées à bien. Par exemple, utiliser un nouvel appareil électronique dans la maison représente une activité nécessitant un apprentissage et une certaine adaptation pour l'habitant.

Maintenant que la notion fondamentale d'activité a été définie, il est nécessaire d'expliquer en quoi elle est au cœur du fonctionnement même des habitats intelligents, puisque l'objectif de ceux-ci est d'offrir à leurs résidents une assistance adaptée grâce à la reconnaissance de leurs activités.

1.3 LA RECONNAISSANCE D'ACTIVITÉS

La reconnaissance d'activités est un domaine qui fait partie de l'Intelligence Artificielle (IA). De nos jours, ce dernier suscite un intérêt toujours plus grandissant chez les chercheurs ; pourtant, sa première définition n'est pas récente. En effet, selon Schmidt *et al.* (1978), la reconnaissance d'activités peut se traduire comme la découverte du but qu'un acteur souhaite atteindre à partir d'une séquence d'actions qu'il est en train de réaliser. Ainsi, la reconnaissance d'activités est la déduction d'une suite d'actions dans l'espace et le temps (la structure d'activité) déterminée et exécutée par une entité qu'un observateur va alors chercher à reconnaître.

Depuis, Roy *et al.* (2013) ont proposé de caractériser la reconnaissance d'activités par la relation qui existe entre l'observateur (*p. ex.* l'habitat intelligent) et l'observé (*p. ex.* le résident). Ainsi, il demeure possible de mettre en place un découpage de la reconnaissance d'activités

en différents types, selon le comportement que l'observé adopte envers l'observateur. Trois comportements distincts ont donc été recensés : positif, négatif ou neutre. Dans le premier cas, l'observé va agir de sorte à faciliter et même collaborer avec l'observateur dans son processus de reconnaissance. Pour ce faire, l'observé peut, par exemple, poser des questions à l'observateur en cas de doutes. Cependant, appliquer un tel type de reconnaissance relève de l'impossible lorsque les résidents des habitats intelligents sont affectés par des troubles cognitifs comme la maladie d'Alzheimer. En effet, la nécessité d'adapter son comportement pour assister l'observateur dans son processus requiert une forte augmentation de la charge cognitive de l'habitant, ce à quoi il leur est pratiquement impossible de faire face. Dans le second cas, où l'observé adopte un comportement négatif envers l'observateur, il va essayer délibérément, et par tous les moyens, d'empêcher le bon déroulement de la reconnaissance. Néanmoins, ceci n'arrive jamais avec les résidents atteints de l'Alzheimer, mais pourrait parfaitement se produire dans le cas d'autres maladies et/ou handicaps. Les personnes atteintes d'Alzheimer entrent plutôt dans la dernière catégorie de comportement, où l'observé n'agit ni pour aider ni pour empêcher le processus de reconnaissance.

La présentation de ces trois types de reconnaissance vient clore la définition générale de la reconnaissance d'activités. Il est maintenant nécessaire de s'intéresser à l'intégration de cette problématique au sein des environnements intelligents.

1.4 LA RECONNAISSANCE D'ACTIVITÉS DANS LES ENVIRONNEMENTS INTELLIGENTS

Depuis sa définition initiale, de nombreux travaux ont fait évoluer la reconnaissance d'activités afin que celle-ci puisse s'adapter au contexte bien spécifique des environnements intelligents (Patterson *et al.*, 2005; Boger *et al.*, 2006; Bouchard *et al.*, 2007; Ghayvat *et al.*, 2018). L'objectif majeur de ces adaptations était de remanier le concept d'environnement ambiant, afin qu'il puisse s'unir avec la problématique de la reconnaissance d'activités.

De ce fait, la première extension à la définition de ce processus appliquée de manière concrète à ces environnements a été décrite par Patterson *et al.* (2005). Leur recherche indique que les observations doivent être réalisées à partir des données fournies par des capteurs de bas niveau. Cette nouvelle vision vient directement s'inscrire dans l'ère de l'informatique ubiquitaire (Weiser, 1991) et devient, par conséquent, beaucoup plus pertinente face à la réalité du problème. L'observé évolue alors dans un environnement composé de différents capteurs, dont le rôle est de récolter les variations qui sont produites par les interactions entre l'acteur et son environnement. Par conséquent, ce sont ces changements qui vont permettre à l'observateur de réaliser le processus de reconnaissance d'activités.

Plus récemment, en se basant sur les travaux précédents de Patterson *et al.* (2005), Roy *et al.* (2013) ont proposé une liste de quatre problématiques pour accomplir le processus de reconnaissance d'activités au sein des environnements intelligents. La première concerne la récolte des valeurs d'un ensemble hétérogène de capteurs de manière uniforme. En d'autres

termes, il s'agit d'offrir la possibilité de recueillir toutes les données produites par l'ensemble de capteurs au travers d'une interface unique, où cette dernière fait totalement abstraction du protocole de communication pour chaque capteur (*p. ex.* RFID, Bluetooth, Wi-Fi, I^2C, *etc.*). Le second défi identifié par Roy *et al.* (2013) est l'interprétation de ces valeurs pour en déduire de l'information intelligible, comme la localisation du résident au sein de l'environnement. Ensuite, la troisième problématique est d'être en mesure de déterminer les actions qui sont effectuées par l'habitant *via* l'interprétation des valeurs renvoyées par les capteurs. Enfin, le dernier défi mentionné par les auteurs concerne l'interprétation de la suite d'actions produites en une activité de plus haut niveau.

Pour répondre à ces quatre problématiques, Roy *et al.* (2013) ont proposé un modèle de reconnaissance d'activités multicouches dont la représentation graphique est donnée en Figure 1.1. Les quatre couches qui composent ce modèle sont indépendantes les unes des autres. Chaque couche a pour objectif de ne répondre qu'à une seule problématique en particulier et doit fournir à la couche supérieure le résultat du traitement achevé. Ainsi, chacune des problématiques peut être traitée séparément.

En définitive, il est donc possible d'effectuer une reconnaissance d'activités au sein d'un environnement intelligent à partir de capteurs de bas niveau—à condition que les différentes problématiques énoncées précédemment soient respectées tout au long du processus.

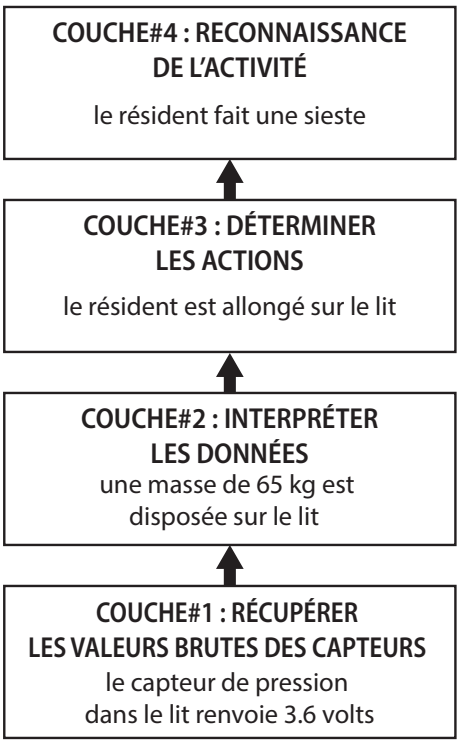


Figure 1.1 : Représentation multicouches de la reconnaissance d'activités dans les environnements intelligents.

1.5 LES HABITATS INTELLIGENTS ET LES *WEARABLE DEVICES*

Au cours des dix dernières années, de nombreux travaux concernant les environnements intelligents et plus particulièrement, les habitats intelligents ont vu le jour. Pour reconnaître les activités des résidents, ceux-ci ont adopté des architectures sensiblement identiques, où les valeurs des capteurs sont centralisées en une entité unique (un serveur) qui va, à elle seule, exécuter toutes les couches de la reconnaissance d'activités (Bouchard *et al.*, 2014; Hu *et al.*, 2016). Néanmoins, certaines distinctions demeurent entre ces différentes implémentations. Par exemple, le Laboratoire d'Intelligence Ambiante pour la Reconnaissance d'Activités (LIARA) (Bouchard *et al.*, 2014) et le Laboratoire de Domotique et informatique Mobile à l'Université

de Sherbrooke (DOMUS) (Giroux *et al.*, 2009) représentent les habitats intelligents hérités du monde de l'industrie. Les autres implémentations d'habitats intelligents comme Gator-Tech (Helal *et al.*, 2005), MavHome (Cook *et al.*, 2003), CASAS (Cook *et al.*, 2013) ou encore Amiqual4home (Lago *et al.*, 2017) reposent toutes, quant à elles, sur des architectures par composants. Cependant, ces habitats intelligents connaissent certaines limitations. En effet, bien que les architectures centralisées simplifient l'utilisation des algorithmes de reconnaissance d'activités, elles ont un impact significatif sur la fiabilité de fonctionnement du système de reconnaissance et donc sur la sécurité du résident qui occupe l'habitat. Bien que de nouveaux travaux essayent de répondre à cette problématique (Cook *et al.*, 2013; Plantevin *et al.*, 2018), les habitats intelligents restent également, encore très dispendieux.

Par ailleurs, les recherches réalisées dans la reconnaissance d'activités au sein d'un habitat intelligent se sont intéressées, en grande partie, à reconnaître les activités d'un unique habitant dans l'environnement (mono-résident) (Vikramaditya Jakkula, 2007; Van Kasteren *et al.*, 2008; Inomata *et al.*, 2009; Ghazvininejad *et al.*, 2011; Belley *et al.*, 2014; Fortin-Simard *et al.*, 2015). Bien que très performantes, ces solutions se sont retrouvées défaillantes lorsque l'habitat est occupé par plusieurs résidents (multi-résident). Ce cas de figure constitue pourtant une problématique bien plus proche de la réalité actuelle des habitats intelligents. De nouvelles techniques de reconnaissance d'activités ont donc été proposées (Crandall et Cook, 2009; Cook et Schmitter-Edgecombe, 2009; Alemdar *et al.*, 2013; Ayuningtyas *et al.*, 2014; Emi et Stankovic, 2015; Mokhtari *et al.*, 2018). Plusieurs d'entre elles (Mihailidis *et al.*, 2004; Tunca *et al.*, 2014) se sont alors tournées vers l'utilisation de *wearable devices*

qui demeurent beaucoup plus accessibles financièrement. En effet, au cours des dernières années, les principaux fabricants de matériel électronique tels que *Garmin*, *Apple*, *Samsung* ou encore *Fitbit* ont contribué à populariser, pour le grand public, la technologie des accessoires connectés à porter sur soi (Figure 1.2). Selon le rapport réalisé par Nielsen (2014), 70% des consommateurs interrogés connaissent cette technologie et 15% d'entre eux se servent d'un *wearable device* dans leur vie de tous les jours en plus de leur téléphone intelligent.

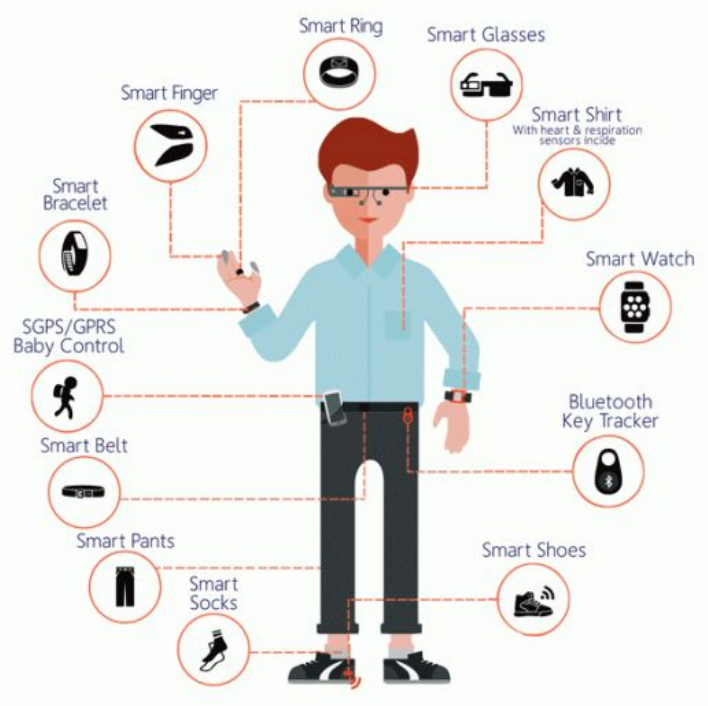


Figure 1.2 : Les différents *wearable devices* (Huifeng *et al.*, 2020).

D'autre part, grâce à la miniaturisation de ces appareils, l'exploitation de nouveaux types de capteurs a été rendue possible, comme les capteurs cardiaques qui n'étaient pas présents dans les téléphones intelligents. En ce sens et puisqu'ils représentent un moyen pratique et

portable pour enregistrer des données physiologiques, les *wearable devices* ont été rapidement et massivement adoptés. (NPD Group, 2015). De plus, ceux-ci tendent à être de moins en moins considérés comme intrusifs par leurs utilisateurs (Gaskin *et al.*, 2017). Ainsi, en plus de la reconnaissance d'activités mono-résident et multi-résidents, ces capteurs portables ont permis l'ouverture de nouveaux horizons de recherche et sont principalement utilisés pour la surveillance de la santé, la réhabilitation physique, la détection de chutes, *etc.* (Patel *et al.*, 2012; Mukhopadhyay, 2014; Delahoz et Labrador, 2014).

Néanmoins, l'augmentation de l'utilisation des *wearable devices* au sein des habitats intelligents a fait émerger de nouvelles problématiques. Dans un premier temps, puisque ceux-ci se trouvent placés sur les résidents, il devient alors possible d'envisager de nouveaux types de reconnaissance permettant d'améliorer la qualité de l'assistance proposée. Cependant, puisque les différentes architectures recensées dans la littérature n'ont pas été initialement prévues pour accueillir ce type de matériel, les *wearable devices* se retrouvent, d'un point de vue matériel, parfois mal intégrés à celles-ci et lorsqu'ils le sont, la fiabilité de la reconnaissance d'activités se retrouve fortement impactée. Il apparaît donc primordial de s'intéresser à une manière de mieux les intégrer aussi bien matériellement que logiciellement afin d'être en mesure de proposer, de manière fiable, un processus de reconnaissance combinée entre les données fournies, à la fois par les capteurs ambiants et portables. Ceci semble pourtant difficilement faisable actuellement tant en termes d'intégration matérielle que d'exploitation logicielle puisque les dispositifs actuels exploitent uniquement les données qu'ils génèrent eux-mêmes.

1.6 DÉFINITION DU PROJET DE RECHERCHE

REVOIR CETTE SECTION

Dans un premier temps, cette thèse propose de s'intéresser à de nouvelles méthodes d'exploitation des *wearable devices* au sein des maisons intelligentes afin d'offrir une meilleure assistance à leurs résidents. Dans un second temps, ce travail propose également la mise en place d'un nouveau type d'architecture pour ces habitats offrant plus une meilleure fiabilité ainsi qu'une meilleure évolutivité matérielle et logicielle. Finalement, afin de réconcilier la cohabitation des capteurs ambiants présents dans les habitats intelligents avec les capteurs *wearable devices*, cette thèse introduit un nouvel outil. Ce dernier a pour principal objectif de permettre l'exploitation de différents types de données issus de sources hétérogènes dans le processus d'apprentissage. Cependant, il vise également à simplifier le déploiement de nouveaux composants logiciels ainsi qu'à favoriser la réutilisation de ces composants. Plus particulièrement, cette thèse permettra de répondre aux questions suivantes :

1. Quels sont les nouveaux apports, en matière d'intelligence, que les *wearable devices* vont permettre de proposer aux résidents des habitats intelligents afin d'améliorer l'assistance qui leur est requise ?
2. Comment faire évoluer les architectures de maisons intelligentes pour leur permettre de mieux s'adapter aux divers types de capteurs (ambiants et *wearable devices*) tout en garantissant un excellent niveau de fiabilité dans l'accomplissement des différents processus d'apprentissage ?

3. Comment prendre en considération la diversité des composants logiciels, et plus précisément, ceux exploités par les *wearable devices*, qui composent les différents processus d'apprentissage en facilitant leur intégration, leur réutilisation ainsi que leur déploiement au sein de l'architecture ?

Cette thèse s'articule autour de trois contributions principales qui font l'objet des publications scientifiques suivantes :

1. Un article de conférence intitulé : « Une méthode indépendante de la position pour la reconnaissance des types de sol à partir de données inertielles provenant d'un *wearable device* » présenté à la 14^{e} conférence internationale IEEE : *Ubiquitous Intelligence and Computing (UIC)* qui s'est déroulée en août 2017 à San Francisco aux États-Unis (Thullier *et al.*, 2017).
2. Un article de conférence intitulé : « Comparaison des méthodes d'acquisition de données inertielles pour une reconnaissance des types de sol indépendante de la position » présenté à la 15^{e} conférence internationale IEEE : *Ubiquitous Intelligence and Computing (UIC)* qui s'est déroulée en octobre 2018 à Guangzhou en Chine (Thullier *et al.*, 2018).
3. Un article de journal intitulé : « LE2ML : un *workbench* d'apprentissage machine basé sur les microservices au sein d'une architecture agnostique, fiable et évolutive pour les maisons intelligentes » soumis et en cours de révision dans le journal : *Journal of Ambient Intelligence and Humanized Computing*.

1.7 ORGANISATION DU DOCUMENT

REVOIR CETTE SECTION

La suite de cette thèse se découpe en sept chapitres principaux. Le Chapitre 2 présente, dans un premier temps, les différentes architectures d'habitats intelligents existantes. Dans un second temps, ce chapitre explique plus en détail le processus de reconnaissance d'activités et plus spécifiquement, les différents algorithmes qui sont utilisés pour l'apprentissage et la classification. Ensuite, le Chapitre 3 explicite le fonctionnement des *wearable devices* ainsi que les différentes applications de ceux-ci au sein des habitats intelligents. Le Chapitre 4 présente une première contribution issue de ce projet de recherche. Dans un premier temps, ce chapitre se concentre sur la description de la solution proposée pour réaliser une reconnaissance des sols par le biais d'un *wearable device*. Dans un second temps, l'ensemble des résultats obtenus lors des différentes phases d'expérimentation sont détaillés et discutés. Par ailleurs, le Chapitre 5 propose une nouvelle architecture en *cluster* pour les maisons intelligentes et le Chapitre 6 introduit LE2ML (LIARA Environment for Modular Machine Learning), un *workbench* modulaire pour l'apprentissage machine, dont la conception repose principalement sur l'architecture proposée. Enfin, le Chapitre 7 clôture cette thèse en tirant les conclusions de chacun des travaux présentés dans cette thèse. De plus, ce chapitre propose différentes perspectives d'améliorations qui pourront être adressées dans le futur.

CHAPITRE II

L'INTELLIGENCE AMBIANTE AU SEIN DES HABITATS INTELLIGENTS

Dans le chapitre précédent, la reconnaissance d'activité a été définie et son application concrète au sein d'environnements intelligents a été brièvement abordée. Néanmoins, avant d'entrer plus en détail sur le fonctionnement de la reconnaissance d'activité, il convient d'étudier les environnements dans lesquels elle est réalisée. Ainsi, ce chapitre commence par présenter l'état de l'art des différents types d'architectures existantes. Ensuite, dans une seconde partie, ce chapitre propose d'examiner le processus de la reconnaissance d'activités de manière approfondie. L'ensemble des étapes préalables qui sont nécessaires au processus d'apprentissage pour la reconnaissance des activités, les différents algorithmes utilisés ainsi que les principales métriques employées pour mesurer de la performance de la reconnaissance y sont notamment présentés. Par ailleurs, certains de ces mécanismes peuvent être intégrés dans des outils (*workbench*) qui permettent, à eux seuls, d'effectuer une importante partie du processus d'apprentissage. Leur facilité d'utilisation a donc mené plusieurs travaux de recherche académique à s'appuyer sur ces outils pour proposer des méthodes de reconnaissance d'activités à l'intérieur des habitats intelligents (Maurer *et al.*, 2006; Shoaib *et al.*, 2013; Ramirez-Prado *et al.*, 2019). Ainsi, dans une dernière partie, ce chapitre présente les trois principaux *workbench* d'apprentissage machine open-source qui offrent le plus de possibilités, mais qui sont également les plus cités dans la littérature.

2.1 LES HABITATS INTELLIGENTS EXISTANTS

Dans la dernière décennie, plusieurs implémentations d'habitats intelligents, visant à mettre en pratique le concept de reconnaissance d'activités multicouche tel que présenté par Roy *et al.* (2013), ont émergé (Cook *et al.*, 2003; Helal *et al.*, 2005; Giroux *et al.*, 2009; Cook *et al.*, 2013; Bouchard *et al.*, 2014; Lago *et al.*, 2017). Néanmoins, en l'absence de toute spécification formelle d'une architecture optimale pour ces habitats, chacune des applications proposées demeure différente malgré leur motivation commune. Puisque l'un des objectifs de cette thèse est de proposer une meilleure intégration des *wearable devices* au sein des habitats intelligents—il convient d'identifier les réelles différences entre ces architectures afin d'en extraire leurs avantages et leurs inconvénients. Pour ce faire, cette première partie se découpe en quatre sous-sections qui correspondent chacune à un type d'implémentation parmi les plus avancées dans le domaine, soit les architectures industrielles avec le LIARA (Bouchard *et al.*, 2014) et le DOMUS (Giroux *et al.*, 2009), les architectures basées composants avec MavHome (Cook *et al.*, 2003), Gator-Tech (Helal *et al.*, 2005) et Amiqual4home (Lago *et al.*, 2017) suivis de CASAS (Cook *et al.*, 2013), l'habitat intelligent qui repose sur un réseau maillé ZigBee et enfin de l'architecture distribuée proposée par Plantevin *et al.* (2018) qui repose sur l'utilisation de transducteurs intelligents. Cependant, cette thèse ne tient pas compte des habitats intelligents qui reposent sur l'utilisation de technologies considérées trop intrusives pour reconnaître les activités de leurs résidents, comme les caméras ou les microphones (Brumitt *et al.*, 2000; Vacher *et al.*, 2011).

2.1.1 TECHNOLOGIES INDUSTRIELLES

Le LIARA (Bouchard *et al.*, 2014) et le Laboratoire DOMUS (Giroux *et al.*, 2009) sont deux environnements académiques de recherche pour la reconnaissance d'activités. Ces deux laboratoires disposent d'habitats intelligents qui reposent sur des technologies héritées du milieu industriel. Comme illustré à la Figure 2.1, leur architecture s'articule essentiellement autour d'une entité de calcul centrale (le serveur) ainsi qu'un automate industriel qui permet de récupérer les valeurs des différents capteurs qui sont divisés en îlots. Ces valeurs sont ensuite enregistrées dans une base de données relationnelle. Ainsi, les couches supérieures de la reconnaissance d'activités peuvent être réalisées sur le serveur par le biais d'une unique interface.

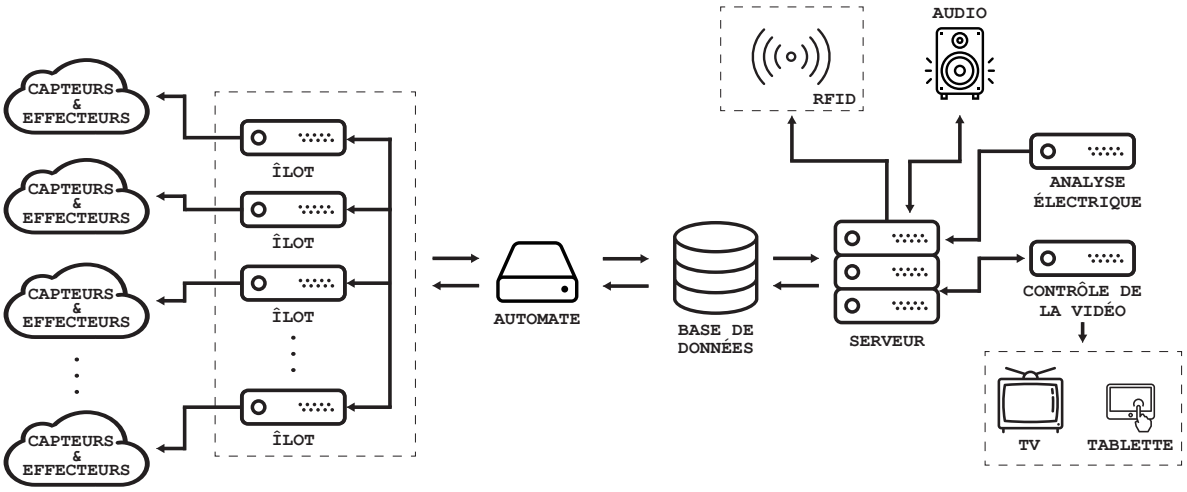


Figure 2.1 : Architecture du LIARA (Bouchard *et al.*, 2014) et du laboratoire DOMUS (Giroux *et al.*, 2009).

Le principal avantage d'une telle architecture concerne l'exploitation de matériel industriel directement. En effet, ce dernier étant conçu pour fonctionner sans interruption dans des environnements difficiles comme des usines, il demeure extrêmement robuste et fiable sur le long terme. Cela permet donc de garantir la qualité de fabrication pour l'habitat intelligent. Aussi, le choix d'une architecture centralisée favorise un accès simplifié aux données de l'habitat. La couche matérielle est abstraite pour les différents programmes de reconnaissance et d'assistance qui peuvent alors interroger directement la base de données.

Malgré ces avantages, une architecture comme celle du LIARA et du laboratoire DOMUS présentent des inconvénients majeurs. Le premier concerne le coût global. En effet, Plantevin (2018) a évalué le prix de ces habitats à 13 500 dollars US sans compter les capteurs, les effecteurs, l'installation et le support. Un investissement de cette ampleur est donc conséquent pour les personnes en perte d'autonomie, car dans une grande majorité, ceux-ci vivent dans des conditions financières précaires, et ce, malgré la prise en charge de certains frais par leur régime d'assurance (Alzheimer's Association, 2018). De plus, bien que le matériel industriel soit un gage de qualité de l'habitat, il n'est pas impossible qu'une panne survienne, par exemple la perte d'un îlot, de l'automate ou du serveur. De ce fait, puisque l'architecture est centralisée, le fonctionnement de l'habitat peut être partiellement ou totalement altéré. Dans le pire des cas, l'assistance au résident devient donc impossible et des situations dangereuses pour sa sécurité peuvent survenir. Finalement, les architectures monolithiques telles que celles utilisées au sein du LIARA et du DOMUS souffrent essentiellement de problèmes d'évolutivité et ne facilitent pas l'intégration de nouveaux capteurs qui doivent

être ajoutés manuellement dans le système. De plus, puisque ces implémentations ont été principalement pensées pour exploiter des capteurs ambiants, l'intégration de capteurs qui ne peuvent être reliés par câble aux îlots de l'habitat demeure extrêmement complexe (Plantevin, 2018). Il apparaît donc clairement ces architectures ne sont pas compatibles avec les *wearable devices* d'un point de vue du matériel ce qui impacte également leur prise en charge par les applications relatives à la reconnaissance d'activités.

2.1.2 OPEN SERVICES GATEWAY INITIATIVE

Alternativement aux architectures industrielles, d'autres architectures qui reposent, quant à elles sur des composants logiciels ont été développées afin de faciliter le besoin d'évolutivité induit par les maisons intelligentes. Le premier exemple d'habitat exploitant ce type d'architecture est MavHome (Cook *et al.*, 2003). Cette dernière exploite sur une interface Common Object Request Broker Architecture (CORBA) [1] permettant la liaison entre les différents composants logiciels et le contrôleur électrique des appareils présents dans l'habitat. Cependant, en raison de la complexité des contraintes imposées par ce standard, des solutions plus récentes comme Gator-Tech (Helal *et al.*, 2005), Amiqual4home (Lago *et al.*, 2017) ou encore les initiatives proposées par Novák et Binas (2011) et Cheng *et al.* (2012) ont préféré l'emploi de la technologie Open Services Gateway initiative (OSGi) [2]. Puisque tous ces habi-

1. www.corba.org
2. www.osgi.org/developer/architecture

tats partagent des caractéristiques identiques, seuls les cas de Gator-Tech et d'Amiqual4home seront détaillés dans cette section, car ils reviennent le plus fréquemment dans la littérature.

En proposant l'habitat Gator-Tech Helal *et al.* (2005) ont montré qu'il est possible de créer une architecture d'habitat intelligent à faible coût, où l'intégration de nouveaux capteurs ambiants demeure simple. Pour ce faire, cette architecture dépend essentiellement de la technologie OSGi. Comme illustré par la Figure 2.2a, chaque capteur de l'habitat possède son propre pilote qui lui permet de communiquer avec l'intégralité du système. Ce pilote est stocké dans une mémoire morte dans le but de conserver les données, même lorsque le matériel n'est pas alimenté. Ainsi, lors de la mise en fonction d'un capteur, celui-ci s'enregistre de manière autonome auprès d'une définition de service. Cette dernière va alors servir de couche d'abstraction pour la création de services de base qui peuvent soit permettre de consommer des données fortement abstraites, soit être combinés afin d'obtenir un service composite. Un service de base peut, par exemple, renvoyer *"la plaque de cuisson est chaude"* lorsque la sonde de température de la cuisinière donne une valeur supérieure à 50°C ; alors qu'un service composite peut, quant à lui, agréger tous les services basiques des capteurs Radio-Frequency IDentification (RFID) pour localiser un résident dans l'habitat. De ce fait, il est possible de concevoir des programmes de reconnaissance et d'assistance sans jamais se préoccuper du protocole de communication des capteurs, ni du format dans lequel ils transmettent les données.

Ce même fonctionnement est utilisé par l'habitat intelligent Amiqual4home. Cet environnement s'appuie sur open Home Automation Bus (openHAB) [3], un *middleware open source* dédié aux habitats intelligents, qui s'appuie sur un environnement OSGi ainsi que sur le *framework* Eclipse SmartHome [4] spécifiquement conçu pour fonctionner dans cet environnement. Ainsi, comme illustré par la Figure 2.2b, openHAB permet de créer une couche d'abstraction autour des couches service, connaissance et logicielle visibles sur la Figure 2.2a.

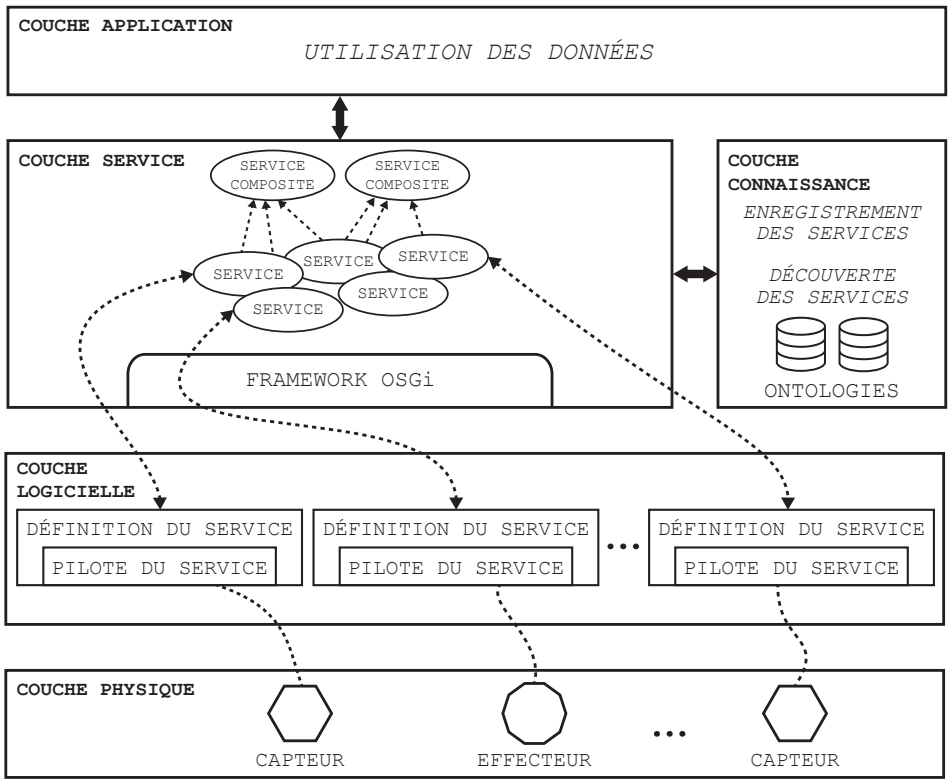
Le principal avantage de ce type d'architectures est qu'elles supportent parfaitement la mise à l'échelle de l'environnement dans le cas d'ajouts ou de remplacements de capteurs ambiants. De plus, l'exploitation de données fortement abstraites facilite le développement de programmes de reconnaissance et d'assistance, puisqu'un raisonnement sur ce type de données demeure plus simple que d'exploiter les données brutes directement (Helal *et al.*, 2005). Enfin, il est possible que le coût d'un tel habitat soit considérablement réduit en comparaison à un habitat basé sur une architecture industrielle. En effet, dans le cas de Gator-Tech, son architecture requiert toujours un serveur central. Néanmoins, elle n'implique pas l'utilisation d'un automate ou d'îlots, et les capteurs utilisés au sein de cet environnement sont basés sur des plateformes à faible coût (Helal *et al.*, 2005).

Cependant, tout comme pour les habitats de type industriels, Gator-Tech et les autres architectures basées composant souffrent du même inconvénient majeur de fiabilité. En effet, le recours à un serveur central comme seule unité de calcul provoque, dans le cas présent, la

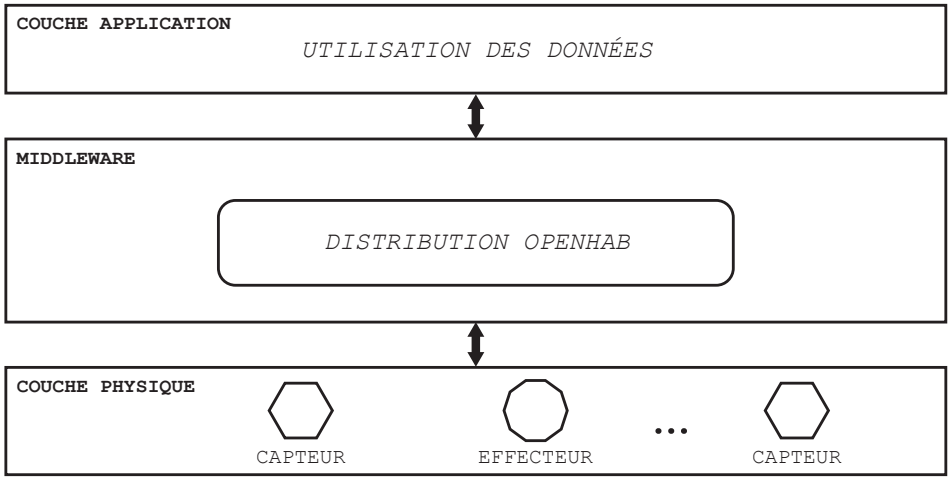
3. www.openhab.org

4. www.eclipse.org/smarthome

création d'un goulot d'étranglement qui peut entraîner un ralentissement général du système lorsqu'un large flux de données transite par le serveur. Ce dernier devient alors surchargé et l'assistance pour le résident n'est plus réalisée convenablement. De plus, bien que l'intégration des *wearable devices* semble possible grâce à la modularité qu'offre OSGi, cette technologie s'avère, en réalité, difficile à mettre en œuvre avec de tels dispositifs, et ce, en raison du manque de flexibilité d'OSGi qui impose un environnement spécifique et particulièrement lourd. De ce fait, cet environnement n'est pas toujours compatible et adapté avec l'utilisation des *wearable devices* qui disposent, la plupart du temps, de plus capacités de calculs et de mémoire.



(a) Gator-Tech



(b) Amiqual4home

Figure 2.2 : Architectures des habitats intelligents Gator-Tech (Helal *et al.*, 2005) et Amiqual4home (Lago *et al.*, 2017).

2.1.3 RÉSEAU MAILLÉ

Grâce aux récentes avancées des technologies de communication sans-fil, de nouveaux types d'habitats intelligents ont émergé. Ainsi, des initiatives telles que CASAS (Cook *et al.*, 2013), ou encore celles proposées par Zhihua (2016) et Zhenyu *et al.* (2011) ont été proposées. Toutes s'articulent autour d'un réseau maillé ZigBee, une technologie de communication sans fil qui sera expliqué plus en détail dans la Figure 3.3.4. Bien qu'ils admettent des différences, ces trois habitats demeurent sensiblement identiques dans la conception de l'architecture qu'ils proposent. De ce fait, cette section traite exclusivement du cas de CASAS, puisque celui-ci reste le plus connus d'entre eux.

En introduisant CASAS, Cook *et al.* (2013) voulaient principalement résoudre le problème du coût que représentent les habitats intelligents, mais également en faciliter leur installation. C'est ainsi que le concept d'habitat intelligent en boîte est né. Comme montré par la Figure 2.3, l'architecture de CASAS se décompose en quatre composants principaux. Tout d'abord, il y a la couche physique qui contient le réseau maillé ZigBee. Dans le cas de cet habitat, il s'agit d'une topologie de réseau sans-fil où tous les nœuds, c'est-à-dire les capteurs et les effecteurs, sont connectés pair à pair et collaborent pour s'échanger des données. Le réseau maillé communique ensuite avec un service de messagerie au travers un pont ZigBee. Ce dernier a pour but de transformer les données des capteurs en messages Extensible Messaging and Presence Protocol (XMPP) de plus haut niveau. Le service de messagerie étant de type *publish/subscribe*, il offre alors la possibilité à d'autres services de s'y connecter. C'est notamment le cas du service d'archivage. Ce dernier récupère, par l'intermédiaire du pont

scribe, certains évènements qui se produisent dans l'habitat afin de les enregistrer dans une base de données d'archives. Finalement, les applications de reconnaissance et d'assistance exploitent les données du *middleware* en les récupérant *via* le pont applicatif.

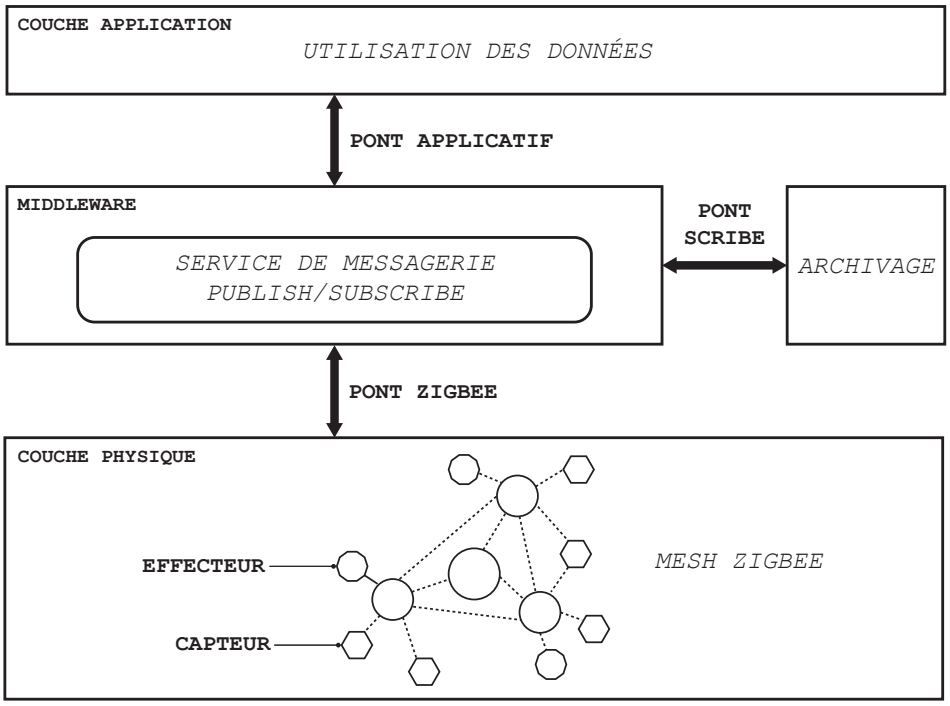


Figure 2.3 : Architecture de l'habitat intelligent CASAS (Cook *et al.*, 2013).

Les habitats intelligents basés sur les réseaux maillés ZigBee sont des solutions réellement moins onéreuses que les autres types d'architectures d'habitats. En effet, Cook *et al.* (2013) rapportent un coût total estimé à 2 765 dollars US, soit presque cinq fois moins qu'un habitat industriel. Par ailleurs, puisque les habitats basés sur un réseau maillé reposent tous sur la technologie de communication sans-fil ZigBee, il semble relativement aisé d'y ajouter de nouveaux capteurs ambiants. Cependant, en ce qui concerne les *wearable devices*, l'utilisation

du protocole ZigBee constitue un inconvénient majeur. En effet, ce type de communication n'étant pas initialement compatible avec les systèmes évolués tels que les ordinateurs ou les téléphones intelligents, les *wearable devices* n'y font pas exception et rares sont ceux qui disposent d'une connectivité ZigBee. La grande majorité d'entre eux ont plutôt adopté la technologie Bluetooth Low Energy (BLE) (Martin, 2014), qui est également devenue compatible avec la topologie réseau maillé récemment (Bluetooth, 2017). Par ailleurs, les architectures comme CASAS souffrent elles aussi de plusieurs points de défaillance qui remettent en question leur fiabilité quant à l'assistance rigoureuse que doivent recevoir les habitants d'une maison intelligente (*p. ex.* les ponts entre les différentes couches).

Il apparaît donc clairement qu'une architecture adaptative pour l'intégration des *wearable devices* est nécessaire. Bien que les architectures basées sur un réseau maillé constituent un premier pas en ce sens, la variété des technologies de communication et des protocoles qu'elles peuvent exploiter n'est pas encore pleinement prise en compte dans leur conception.

2.1.4 TRANSDUCTEURS INTELLIGENTS DISTRIBUÉS

Le principal inconvénient partagé par toutes les architectures de maisons intelligentes identifiées dans les sections précédentes réside dans la centralisation de leur conception. En effet, chacune d'entre elles possède au moins un point de défaillance, ce qui remet alors en question la fiabilité de ces habitats. En effet, ces points de défaillance peuvent conduire à un dysfonctionnement partiel ou total du processus de reconnaissance d'activités qui est effectué au sein de ces habitats et ainsi potentiellement compromettre la sécurité de leurs résidents.

Toutefois, cette problématique en particulier a été traitée il y a plusieurs années, dans différents domaines de l'informatique, tels que le calcul distribué, à l'aide de mécanismes de redondance et de partitionnement de données (*clustering*) (Dikaiakos *et al.*, 2009; Zaharia *et al.*, 2010; Jafarnejad Ghomi *et al.*, 2017).

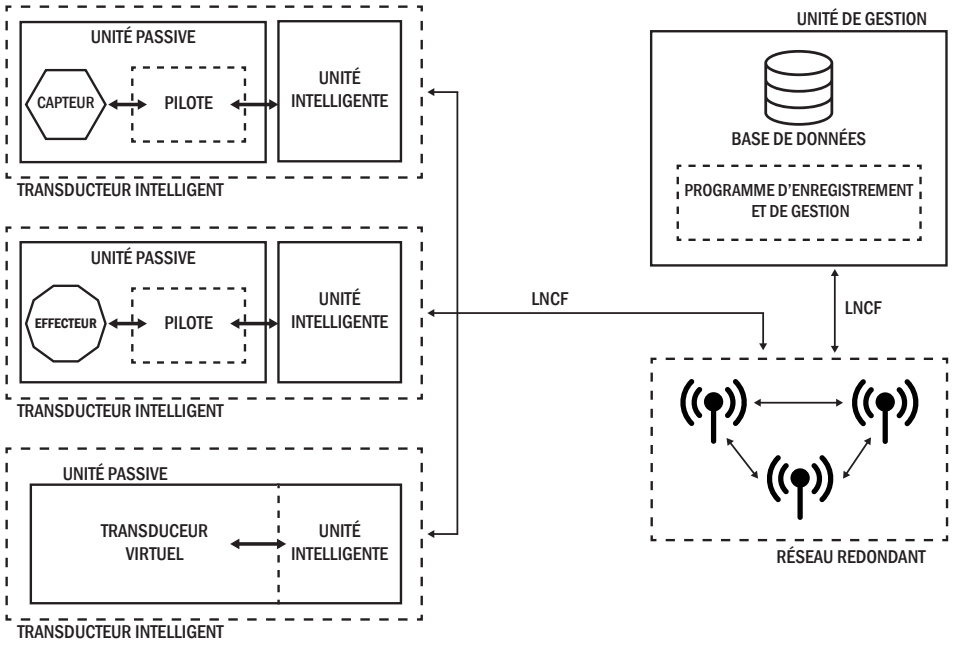


Figure 2.4 : Architecture qui repose sur l'utilisation de transducteurs intelligents (Plantevin *et al.*, 2018).

Dans cette optique, Plantevin *et al.* (2018) ont plus récemment proposé une architecture de maison intelligente distribuée qui n'inclut pas de point de défaillance critique. Pour ce faire, il ont suggéré la mise en place de transducteurs intelligents, définis selon la norme IEEE 1451.4 (Institute of Electrical and Electronics Engineers, 1999). Comme illustré en Figure 2.4, les transducteurs intelligents peuvent être de deux types différents : les transducteurs physiques et les transducteurs virtuels, chacun étant composé de deux unités distinctes, l'unité passive

et l'unité intelligente. Qu'elle soit matérielle ou logicielle, l'unité passive sert à produire des données de haut niveau à partir d'un ensemble d'évènements de bas niveau qui se produisent dans l'environnement extérieur. Par ailleurs, l'unité intelligente, quant à elle, représente l'entité de calcul principale qui fait qu'un transducteur demeure autonome. En effet, son rôle est de communiquer avec le reste de l'environnent et d'exécuter les applications nécessaires pour réaliser le processus de reconnaissance d'activités grâce aux données acquises auprès de l'unité passive. La communication entre ces deux unités est réalisée soit *via* un lien série pour les transducteurs matériels, soit par le biais d'une communication de type *publish/subscribe* basée sur ZeroMQ [5] pour les transducteurs virtuels. Puisque cette architecture repose sur un très grand nombre de transducteurs intelligents, ceux-ci sont tous interconnectés par un réseau sans fil redondant qui leur permet de communiquer entre eux par l'intermédiaire du protocole Light Node Communication Framework (LNCF)—un protocole également introduit par Plantevin *et al.* (2017). Enfin, tous les transducteurs doivent s'enregistrer auprès d'une entité centrale afin de récupérer le pilote ainsi qu'un fichier d'instructions pour leur permettre fonctionner de manière adéquat. Cette opération est effectuée une seule fois, lors du premier démarrage des transducteurs. De ce fait, bien que cette unité de gestion soit nécessaire pour l'installation de nouveaux transducteurs, son rôle est, par la suite, d'assurer le suivi des transducteurs qui composent l'architecture et d'émettre des alertes dans le cas où un transducteur serait en panne.

5. https://zeromq.org

Les deux avantages principaux de cette récente implémentation d'architecture demeurent son faible coût (estimé à 1 950 dollars US) ainsi que son évolutivité vis-à-vis du matériel, puisque la mise en place de nouveaux capteurs y est facilitée. De plus, puisque l'objectif principal étant de supprimer les points de défaillance, cette architecture offre un excellent niveau de fiabilité, car le dysfonctionnement d'un transducteur en particulier n'affecte pas le système dans son intégralité. Il est alors toujours possible de réaliser le processus de reconnaissance d'activité pour offrir l'assistance requise aux résidents de l'habitat. Bien que l'unité de gestion demeure toujours un point central dans la conception de cette architecture, les auteurs ont mentionné que cette entité peut être hébergée dans le *cloud*. Ceci afin de garantir la fiabilité de sa disponibilité en cas de défaillance. De plus, puisque cette unité est indispensable uniquement lors de la première mise en place de l'architecture ou pour ajouter de nouveaux transducteurs, le bon fonctionnement de l'habitat intelligent ne sera pas compromis en cas de panne. En ce sens, il reste possible de réaliser une opération de maintenance sans mettre en danger les habitants de la maison intelligente. Néanmoins, la prise en charge de l'hétérogénéité des composants logiciels qui constituent le processus de la reconnaissance d'activités semble avoir été mise de côté dans la conception de cette architecture. Ainsi, cette architecture semble, de manière générale, manquer d'éléments indispensables qui permettent de favoriser une bonne intégration logicielle des *wearable devices*.

2.2 LE PROCESSUS D'APPRENTISSAGE POUR LA RECONNAISSANCE DES ACTIVITÉS

Le processus d'apprentissage pour la reconnaissance d'activités est un ensemble d'étapes qu'il est généralement possible de réaliser pour reconnaître des activités. Tel qu'illustré par la Figure 2.5, ce processus requiert des données initiales fournies par les capteurs ambiants ou portables. Celles-ci n'ont alors été soumises à aucun traitement en particulier, ce sont donc des données brutes. Principalement en fonction de la condition des capteurs et de la transmission des valeurs, les données obtenues peuvent se retrouver altérées. Afin d'améliorer leur qualité, il est donc possible, dans un second temps, d'effectuer une étape de prétraitement. Ensuite, puisque les données renvoyées représentent un signal ayant une longueur potentiellement infinie, il est impératif de segmenter ce dernier en plusieurs fragments, sur chacun desquels des propriétés discriminantes sont calculées. L'ensemble de celles-ci, une fois extraites, constitue alors une abstraction du fragment traité. Le signal fragmenté est donc réduit à un unique

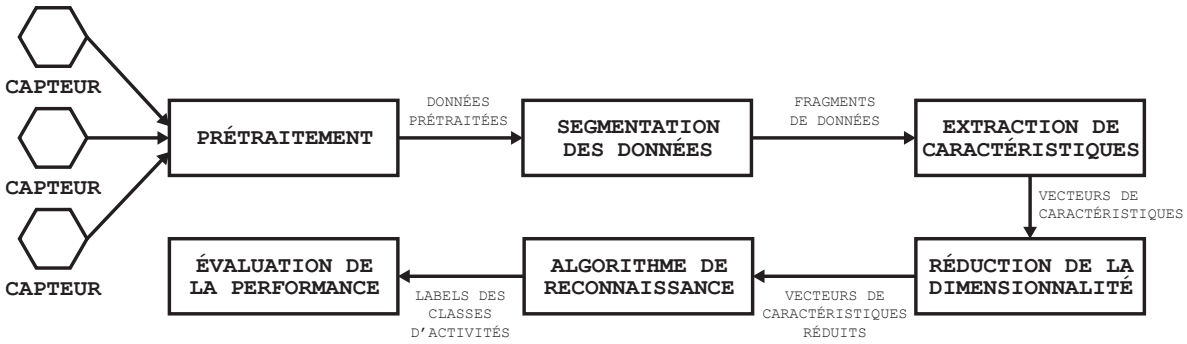


Figure 2.5 : Processus d'apprentissage pour la reconnaissance d'activités.

vecteur de caractéristiques pour chaque segment. Dans certains cas, il peut s'avérer que celui-ci contienne un nombre de propriétés suffisant pour impacter négativement la performance de la reconnaissance—auquel cas, une étape supplémentaire pour réduire la dimensionnalité du vecteur est requise. Cet ensemble de données est ensuite utilisé comme entrée d'un algorithme d'apprentissage qui, pour chacun des vecteurs réduits, retournera les noms des classes associées aux activités à reconnaître. Finalement, la dernière étape du processus consiste en une évaluation de la performance de la reconnaissance.

2.2.1 LES ÉTAPES PRÉLIMINAIRES

Avant d'être en mesure d'utiliser les algorithmes d'apprentissage, il est nécessaire que les données brutes issues des capteurs passent par un ensemble d'étapes préliminaires pour être transformées en caractéristiques discriminantes. Ainsi, cette section présente les méthodes les plus utilisées en ce qui concerne le prétraitement et la segmentation des données, mais également l'extraction de caractéristiques et la réduction du nombre qui en est produit, c'est-à-dire la réduction de la dimensionnalité.

PRÉTRAITEMENT DES DONNÉES

Le prétraitement des données est la première étape du processus d'apprentissage pour la reconnaissance d'activités. En effet, après l'acquisition des valeurs renvoyées, autant par les capteurs ambiants que les capteurs portables présents au sein d'un environnement intelligent,

plusieurs facteurs peuvent influer sur la qualité de ces données. Par exemple, un mauvais fonctionnement du matériel ou un problème de transmission comme une latence réseau peuvent entraîner l'acquisition de données incohérentes, erronées ou encore partiellement manquantes. L'objectif de cette étape est donc d'améliorer les données récoltées en réduisant le bruit, ou en filtrant les éléments nuisibles, tout en essayant de conserver les caractéristiques dynamiques essentielles du signal.

Dans le cas où les données comprendraient des valeurs partiellement manquantes, plusieurs méthodes peuvent être employées pour compenser cette perte. Dans un premier temps, il est tout simplement possible d'ignorer les données incriminées. Néanmoins, cette méthode n'est pas pertinente lorsque l'absence d'information est très élevée. De ce fait, il est possible de remplacer les valeurs manquantes pour un attribut donné grâce à une estimation qui peut être déterminée selon deux procédés différents. Le premier consiste à calculer la moyenne de toutes les valeurs non manquantes pour l'attribut en question. La seconde méthode implique que les données soient étiquetées et consiste à remplacer la valeur manquante d'un attribut par la moyenne des valeurs présentes qui appartiennent à la même classe celui-ci.

De plus, il est admis que les capteurs RFID et les accéléromètres sont les deux types de capteurs (ambiant et portable) les plus sujets au bruit. Or, de nombreux travaux se basent sur l'exploitation de ceux-ci dans le but de reconnaître des activités (Ravi *et al.*, 2005; Stikic *et al.*, 2008; Buettner *et al.*, 2009; Khan, 2011; Mannini *et al.*, 2017). De ce fait, Wang *et al.* (2011) ont comparé quatre filtres différents ayant pour but de réduire le bruit dans les données générées par un accéléromètre, soit : un filtre médian (Huang *et al.*, 1979), un filtre passe-bas

de type *Butterworth* (Butterworth, 1930), une transformée en ondelettes discrète spécifique (Discrete Wavelet Package Decomposition (DWPD)) (Mallat, 1989) et un filtre de Kalman (Welch et Bishop, 2006), où ce dernier s'est révélé comme le plus efficace. Abreu *et al.* (2014) ont, quant à eux, montré que le filtre de Kalman était également le plus performant pour réduire le bruit présent dans le leur système de localisation intérieure basé sur des capteurs RFID.

SEGMENTATION DES DONNÉES

Dans le processus d'apprentissage, la segmentation des données peut intervenir, soit après la première phase de prétraitement, soit directement sur les données brutes si celles-ci ne sont pas altérées. Cette opération consiste à découper un signal donné en plusieurs fragments plus petits. En d'autres termes, le signal est morcelé en une séquence de segments qui sont délimités par un temps de départ et un temps d'arrivée. Du point de vue du traitement de signal, cette opération est appelée le fenêtrage. Ainsi, il est possible d'observer le signal d'origine $x(t)$ de longueur théoriquement infinie, sur une durée finie T en le multipliant par une fonction fenêtre d'observation $w(t)$ tel que,

$$x_w(t) = x(t)\, w(t), \quad 0 \leq t \leq T-1. \tag{2.1}$$

En ce qui concerne la reconnaissance d'activités, il est possible de regrouper les techniques de segmentation en trois groupes distincts : la segmentation basée sur les activités, la

segmentation basée sur des évènements et la segmentation par fenêtre glissante (Banos *et al.*, 2014).

La segmentation basée sur les activités revient à partitionner le signal selon la détection d'un changement significatif correspondant à une activité. Un tel découpage peut-être réalisé grâce à l'analyse du domaine fréquentiel du signal ou de son énergie (Sekine *et al.*, 2000; Guenterberg *et al.*, 2009).

La segmentation basée sur les évènements exploite, quant à elle, un découpage du signal selon la manifestation d'évènements spécifiques. Par exemple, cette méthode est souvent utilisée dans la reconnaissance de la marche ou d'autres activités similaires comme courir, monter ou descendre des escaliers, *etc.* Puisque ces activités sont cycliques et requièrent des actions inévitables (*p. ex.,* le contact du pied sur le sol), il est possible de segmenter le signal par l'analyse de l'accélération (Sant'Anna et Wickström, 2010), mais également en exploitant d'autres types de données comme celles issues de capteurs de pression (Crea *et al.*, 2012).

Enfin, la segmentation par fenêtre glissante peut se traduire comme étant la translation d'une fenêtre sur l'axe temporel du signal, où chaque déplacement de la fenêtre représente un segment. Une fenêtre est définie par une fonction mathématique, une taille ainsi qu'un pourcentage de superposition. En ce qui concerne ce dernier, si une fenêtre est définie comme ayant une taille de 100 secondes avec un pourcentage de superposition de 50%, le premier intervalle sera $[0, 100]$ tandis que le second sera $[50, 150]$ et ainsi de suite. Parmi les fonctions de fenêtrage les plus fréquemment utilisées, il est possible de mentionner, la fenêtre de

Hamming ($w(t)_{Ham}$), celle de Kaiser ($w(t)_K$), de Hann ($w(t)_{Han}$) ou encore la fenêtre de Blackman ($w(t)_B$) respectivement exprimées par :

$$w(t)_{Ham} = \begin{cases} 0.54 - 0.46\cos(\frac{2\pi t}{T-1}), & \text{si } 0 \leq t \leq T-1 \\ 0, & \text{sinon.} \end{cases} \tag{2.2}$$

$$w(t)_K = \begin{cases} \frac{I_0\left[\alpha\sqrt{1-(\frac{2t}{T-1}-1)^2}\right]}{I_0[\alpha]}, & \text{si } 0 \leq t \leq T-1 \\ 0, & \text{sinon.} \end{cases} \tag{2.3}$$

$$w(t)_{Han} = \begin{cases} 0.5 - 0.5\cos(\frac{2\pi t}{T-1}), & \text{si } 0 \leq t \leq T-1 \\ 0, & \text{sinon.} \end{cases} \tag{2.4}$$

$$w(t)_B = \begin{cases} 0.42 - 0.5\cos(\frac{2\pi t}{T-1}) + 0.08\cos(\frac{4\pi t}{T-1}), & \text{si } 0 \leq t \leq T-1 \\ 0, & \text{sinon.} \end{cases} \tag{2.5}$$

où, I_0 est la fonction de Bessel modifiée d'ordre zéro, α est le paramètre réel non nul caractérisant la forme de la fenêtre et T est la taille de la fenêtre. Néanmoins, tous les paramètres impliqués dans le processus de la segmentation doivent être déterminés en fonction de la problématique, où l'objectif est de trouver le meilleur compromis entre la rapidité d'exécution et la performance de reconnaissance (Banos *et al.*, 2014).

EXTRACTION DE CARACTÉRISTIQUES

L'extraction de caractéristiques est une phase importante dans le processus d'apprentissage pour reconnaître des activités. L'objectif de cette opération consiste à produire un vecteur de caractéristiques discriminantes pour chaque fragment du signal généré, grâce à un certain nombre de fonctions mathématiques. Il existe deux principaux domaines auxquels appartiennent les différentes techniques d'extraction : le domaine temporel et le domaine fréquentiel (Huynh et Schiele, 2005; Figo *et al.*, 2010; Cleland *et al.*, 2013). Les caractéristiques issues du domaine temporel demeurent les plus simples à obtenir. Il est notamment possible d'y retrouver des métriques statistiques comme la moyenne, la variance, l'écart type, l'asymétrie ou encore l'aplatissement du signal, mais également des métriques dites « enveloppe » comme la médiane, le minimum ou le maximum du signal. De plus, de nombreuses autres métriques temporelles peuvent être utilisées dans la reconnaissance d'activités, comme la corrélation entre deux signaux ou encore le *zero crossing rate*—qui correspond au taux de changement de signe d'un signal. Les formules de l'asymétrie S, de l'aplatissement K et de la corrélation ρ de deux signaux n et n' sont respectivement données par :

$$S(n) = \frac{N}{(N-1)(N-2)} \cdot \left(\sum_{i=1}^{N} \frac{(n_i - \bar{n})^3}{\sigma_n^3} \right) \tag{2.6}$$

$$K(n) = \frac{N(N+1)}{(N-1)(N-2)(N-3)} \cdot \left(\sum_{i=1}^{N} \frac{(n_i - \bar{n})^4}{\sigma_n^4} - \frac{3(N-1)^2}{(N-2)(N-3)} \right) \tag{2.7}$$

$$\rho\left(n,n'\right)=\frac{cov\left(n,n'\right)}{\sigma_n\cdot\sigma_{n'}} \tag{2.8}$$

où N représente la taille d'un fragment du signal, n_i est le $i^{ème}$ élément du fragment, $\bar{n}$ est la valeur moyenne du fragment et σ_n correspond à l'écart type du fragment.

D'autre part, afin d'exploiter des caractéristiques issues du domaine fréquentiel, il est nécessaire de réaliser un passage vers celui-ci, depuis le domaine temporel. Pour ce faire, il s'agit de calculer la transformée de Fourier discrète du signal. Dans la grande majorité des cas, c'est l'algorithme de la transformée de Fourier rapide (Fast Fourier Transform ou FFT) qui est utilisé à cet effet (Brigham et Morrow, 1967). Malgré sa popularité, il est également possible d'utiliser l'algorithme de Goertzel pour réaliser la transformation depuis le domaine temporel vers le domaine fréquentiel (Sysel et Rajmic, 2012). Dès lors, les caractéristiques fréquentielles qui peuvent être calculées sont, la composante continue (*DC Component*), l'énergie spectrale E et l'entropie H telles que,

$$E=\frac{1}{N}\cdot\left(\sum_{i=1}^{N}\mid n_i\mid\right) \tag{2.9}$$

$$H=-\sum_{i=1}^{N}P_i\ln P_i \tag{2.10}$$

où n représente la transformée fréquentielle d'un signal de taille N obtenue grâce à la FFT, n_i est le $i^{ème}$ élément du signal et P_i est la probabilité de n_i donnée par l'équation suivante :

$$P_i = \frac{n_i}{N - E} \tag{2.11}$$

En dernier lieu, Mitchell *et al.* (2013) ont proposé une méthode de reconnaissance d'activités qui repose sur l'analyse par ondelettes (Discrete Wavelet Transform ou DWT). À l'inverse de la transformée de Fourier, cette technique permet d'obtenir des caractéristiques à la fois issues du domaine temporel et fréquentiel, par décomposition d'un signal en un ensemble de coefficients d'ondelettes.

RÉDUCTION DE LA DIMENSIONNALITÉ

La dernière étape préliminaire concerne la réduction de la dimensionnalité du vecteur de caractéristiques. Cette étape est importante dans la mesure où elle permet de sélectionner l'ensemble des valeurs qui seront conservées pour pallier le *fléau de la dimension* (Bellman, 1957). Ce phénomène intervient lorsque la taille du vecteur de caractéristiques croît de manière conséquente. Les données peuvent alors se retrouver isolées et deviennent éparses, ce qui a pour effet de dégrader l'efficacité de la reconnaissance d'activités. Ainsi, en réduisant la dimensionnalité de ces données, l'objectif est de conserver les caractéristiques pertinentes tout en minimisant la perte d'information. Bien que ce processus doive permettre une meilleure

reconnaissance, l'exploitation d'un plus petit vecteur peut également induire un traitement postérieur plus rapide.

Actuellement, il existe une quantité importante de techniques de réduction de dimensionnalité. Par conséquent, Van Der Maaten *et al.* (2009) ont proposé une taxonomie dans laquelle ils recensent l'ensemble des méthodes existantes. En premier lieu, il possible de retrouver des techniques de réduction de la dimensionnalité orientées vers la sélection de caractéristiques qui reposent sur des évaluations statistiques des données, comme la covariance ou l'entropie. L'avantage majeur de ces méthodes réside dans leur simplicité de mise en œuvre. D'autre part, bien qu'elle demeure une technique plus complexe, l'analyse par composantes principales (Principal Component Analysis ou PCA), a souvent été utilisée dans le domaine de la reconnaissance d'activités (He et Jin, 2009; Altun et Barshan, 2010; Chen *et al.*, 2012; Leightley *et al.*, 2013). Elle permet, de transformer des caractéristiques qui sont initialement corrélées en de nouvelles caractéristiques décorrélées les unes des autres, les composantes principales.

Le nombre de déclinaisons d'algorithmes de sélection de caractéristiques étant considérable, le choix quant à la méthode à adopter n'est pas trivial. En effet, celui-ci doit être fait en fonction de la problématique à traiter et l'algorithme adéquat est bien souvent déterminé de manière empirique.

2.2.2 LES ALGORITHMES D'APPRENTISSAGE

Dans la section précédente, les différents processus de prétraitement ont été présentés. Ces étapes sont importantes puisqu'elles permettent de préparer les données afin de raffiner leur qualité avant qu'elles soient utilisées par un algorithme d'apprentissage. Ainsi, reconnaître avec précision les activités d'un résident est l'objectif principal des habitats intelligents afin qu'ils puissent recevoir une assistance la plus adaptée possible. Par conséquent, de nombreuses méthodes algorithmiques issues du domaine de l'intelligence artificielle ont été proposées pour parvenir à reconnaître des activités. Cette section présente donc l'ensemble des différentes méthodes parmi les plus utilisées dans ce domaine.

Les algorithmes d'apprentissage, également qualifiés d'algorithmes de reconnaissance ou de classification, peuvent être décomposés en deux grandes familles : les méthodes supervisées et non supervisées. Un algorithme est dit supervisé si les données recueillies sont divisées en deux sous-ensembles respectivement appelés ensemble d'entraînement et de validation où les deux ensembles de données contiennent des instances étiquetées avec la classe correspondante à une activité donnée. À l'inverse, les algorithmes non supervisés ne requièrent pas l'étiquetage des données. Cependant, il est également possible de rencontrer, à plus faible mesure, des techniques d'apprentissage semi-supervisées qui combinent l'utilisation de données étiquetées et non-étiquetées lors la phase d'entraînement (Zhu, 2005; Chapelle *et al.*, 2006).

LES ALGORITHMES PROBABILISTES

Dans l'univers des algorithmes probabilistes, il est possible de retrouver les méthodes bayésiennes qui constituent des méthodes supervisées probabilistes. Celles-ci s'articulent autour des réseaux bayésiens. Un réseau bayésien est définis par un graphe orienté acyclique dans lequel, chacun des nœuds (sommets) représente une variable aléatoire et chacune des arêtes correspond à la dépendance conditionnelle existante entre un ou plusieurs nœuds du graphe (Heckerman *et al.*, 1995). Pour appliquer ce principe dans un système de reconnaissance d'activités, il s'agit de considérer que les nœuds du graphe correspondent aux actions ou aux activités et que les arêtes sont les liens qui lient des actions aux activités. Ces actions peuvent être conjointes à plusieurs activités. Par conséquent, si la probabilité de chaque activité ainsi que les probabilités conditionnelles des nœuds sont connues, il est alors possible de mettre à jour la probabilité de chacune des activités en fonction des observations faites dans l'environnement.

De manière plus concrète, la Figure 2.6 illustre l'exemple d'un système de reconnaissance s'appuyant sur un réseau bayésien. Celui-ci est en charge de reconnaître lorsqu'un résident est train de réaliser trois activités dans la cuisine d'un environnement intelligent soit, « *faire du thé* » (A_1), « *faire du riz* » (A_2) et « *faire du café* » (A_3). Grâce à l'observation des habitudes de l'habitant, il est possible de définir les probabilités initiales des activités ainsi que les probabilités conditionnelles. Elles sont respectivement données dans la Figure 2.6, au-dessus de chaque activité à reconnaître et sous chacune des actions nécessaires à leur

réalisation. Ensuite, si l'évènement e_2 correspondant à l'action « *sortir le riz* » est observé, les probabilités de chaque activité selon cet évènement $p(A_i|e_2)$ sont calculées en appliquant la formule de Bayes définie par :

$$p(A_i|e_n) = \frac{p(e_n|A_i) \times p(A_i)}{\sum_{j=1}^{J} p(e_j|A_j) \times p(A_j)} \tag{2.12}$$

où $p(A_i)$ est la probabilité initiale de l'activité A_i et $p(e_n|A_i)$ est la probabilités conditionnelles de l'activité A_i selon l'évènement e_n. Grâce aux données fournie par cet exemple, il est donc possible de réaliser l'application numérique suivante à l'aide de l'Équation 2.12 :

$$\begin{aligned} p(A_1|e_2) &= \frac{p(e_2|A_1) \times p(A_1)}{p(e_2|A_1) \times p(A_1) + p(e_2|A_2) \times p(A_2) + p(e_2|A_3) \times p(A_3)} \\ &= \frac{0 \times 0.2}{0 \times 0.2 + 1 \times 0.6 + 0 \times 0.2} = 0 \end{aligned}$$

$$\begin{aligned} p(A_2|e_2) &= \frac{p(e_2|A_2) \times p(A_2)}{p(e_2|A_1) \times p(A_1) + p(e_2|A_2) \times p(A_2) + p(e_2|A_3) \times p(A_3)} \\ &= \frac{1 \times 0.6}{0 \times 0.2 + 1 \times 0.6 + 0 \times 0.2} = 1 \end{aligned}$$

$$\begin{aligned} p(A_3|e_2) &= \frac{p(e_2|A_3) \times p(A_3)}{p(e_2|A_1) \times p(A_1) + p(e_2|A_2) \times p(A_2) + p(e_2|A_3) \times p(A_3)} \\ &= \frac{0 \times 0.2}{0 \times 0.2 + 1 \times 0.6 + 0 \times 0.2} = 0 \end{aligned}$$

Finalement, les probabilités sont mises à jour et il apparaît que l'activité « *faire du riz* » (A_2) est la seule activité possible puisque celle-ci admet une probabilité de 1 lorsque l'évènement « *sortir le riz* » (e_2) est observé. Le système conclut donc que le résident de l'habitat intelligent est en train de préparer une portion de riz.

Les principaux avantages dans l'utilisation des réseaux bayésiens sont, la simplicité de leur implémentation ainsi que leur performance, tant en termes de taux de reconnaissance dans le cas des activités au sein d'un habitat intelligent, qu'en termes de temps d'exécution et de consommation d'espace mémoire (Friedman *et al.*, 1997). Néanmoins, ceux-ci nécessitent de connaître l'ensemble des activités, des actions et des probabilités conditionnelles qui les relient pour fonctionner ; ce qui constitue un inconvénient majeur, car ces informations ne sont pas toujours disponibles.

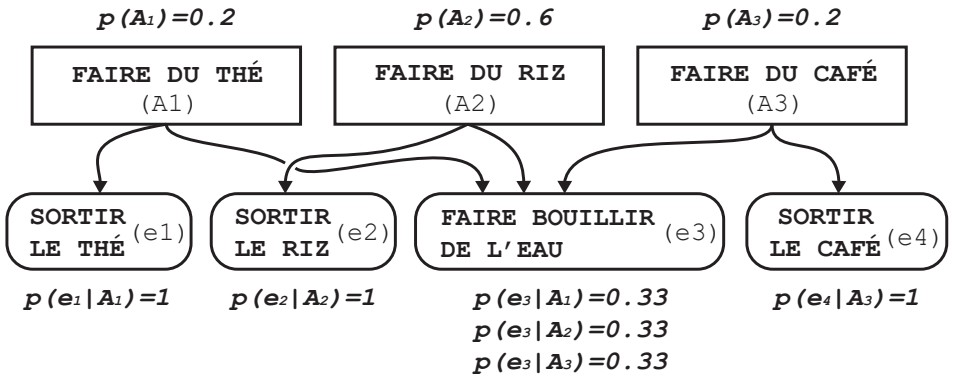


Figure 2.6 : Exemple de réseau bayésien naïf.

Les modèles de Markov cachés (Hidden Markov Models ou HMMs) ainsi que les champs aléatoires conditionnels (Conditional Random Fields ou CRFs) sont d'autres exemples

d'algorithmes statistiques qui ont été utilisés pour la reconnaissance d'activités au sein des habitats intelligents (Oliver *et al.*, 2004; Nazerfard *et al.*, 2010; Van Kasteren *et al.*, 2011). Les HMMs peuvent être représentés comme des automates probabilistes à états finis dont l'évolution au cours du temps est entièrement déterminée par une probabilité initiale et des probabilités de transitions entre états. De plus, l'état du système, soit l'activité à reconnaître, n'est pas directement observable, mais caché par un processus d'observation. Autrement dit, au temps t, le système qui est dans l'état invisible q_t émet l'observation O_t. Ces observations sont, quant à elles, visibles et elles peuvent, par exemple, correspondre à la valeur d'un capteur. Tout comme pour les réseaux bayésiens, les HMMs offrent d'excellentes performances de reconnaissance et d'exécution. Néanmoins, la complexité de leur mise en place et de leur compréhension ainsi que la connaissance initiale requise pour leur fonctionnement sont autant de facteurs limitant leur utilisation à la reconnaissance d'un nombre très restreint d'activités.

Par ailleurs, les CRFs permettent, quant à eux, de modéliser les dépendances entre un ensemble d'observations réalisées sur une séquence et un ensemble d'étiquettes. En comparaison avec un HMM, un CRF ne repose pas sur l'hypothèse forte d'indépendance des observations entre elles, conditionnellement aux états associés (les activités). De plus, ils ne combinent pas de probabilités conditionnelles locales, évitant ainsi une estimation biaisée de ces probabilités si trop peu d'exemples étiquetés sont disponibles. Autrement dit, les CRFs permettent de limiter le nombre de paramètres initiaux requis dans la mise en place d'un HMM, offrant alors une généralisation du modèles de Markov.

LES ARBRES DE DÉCISION

Les arbres de décision appartiennent à la catégorie des méthodes d'apprentissage supervisées. Comme illustré par la Figure 2.7, la classification des données est faite par une suite de choix logiques représentée sous forme d'un arbre, où chacun de ses nœuds possède un unique parent excepté le nœud racine. Ainsi, les feuilles de l'arbre correspondent aux classes des activités à reconnaître et les nœuds non terminaux sont des règles de classification qui doivent être vérifiées grâce aux attributs de la donnée qui est testée.

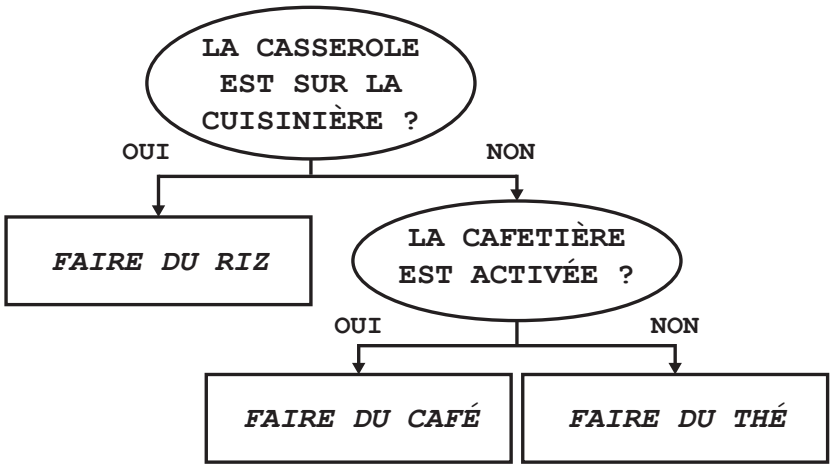


Figure 2.7 : Exemple d'un arbre de décision binaire.

Dans le domaine de la reconnaissance d'activités, les algorithmes d'arbres de décision qui sont les plus fréquemment utilisés sont, Iterative Dichotomiser 3 (ID3) ainsi que son extension, `C4.5` (Quinlan Ross, 1993). Les principales différences entre ces deux algorithmes sont qu'à l'inverse d'ID3, `C4.5` accepte les variables continues en plus des variables discrètes et ce dernier est également capable de gérer les données manquantes de façon automatique.

Enfin, il permet de réduire le risque de sur-apprentissage, qui est très élevé avec l'algorithme ID3, grâce à une technique de *pruning* (Bao et Intille, 2004; Ravi *et al.*, 2005; Tapia *et al.*, 2007). Ce phénomène intervient lorsque le modèle d'apprentissage est trop similaire aux données d'entraînement et donc plus suffisamment générique pour reconnaître correctement de nouvelles activités. Malgré ces différences, le fonctionnement de ces deux algorithmes reste très similaire. Pour chaque attribut de l'ensemble de données qui n'a pas déjà été traité, son entropie H (*cf.* Équation 2.10) est calculée. Ensuite, l'attribut ayant l'entropie la plus faible est sélectionné et une séparation des données est faite en fonction de celui-ci (*p. ex.* « *la casserole est sur la cuisinière* » ou « *la casserole n'est pas sur la cuisinière* »). Enfin, la structure finale de l'arbre est créée de manière récursive en ne considérant uniquement les attributs qui n'ont jamais été sélectionnés, pour chaque sous-ensemble de données. Ceci vient donc conclure la phase d'entraînement d'un arbre de décision. La phase de reconnaissance, quant à elle, ne nécessite qu'un simple parcours de l'arbre. Ainsi, en reprenant l'exemple de la Figure 2.7—si les données de l'instance à reconnaître correspondent au positionnement de la casserole sur la cuisinière de l'habitat intelligent, alors l'activité réalisée par le résident est « *faire du riz* ».

L'utilisation des arbres de décision offre plusieurs avantages. Dans un premier temps, ceux-ci restent relativement simples à implémenter, mais surtout, ils offrent une très grande facilité de compréhension. De plus, ils ont démontré une excellente performance tant en termes de rapidité d'exécution qu'en termes de taux de reconnaissance (Bao et Intille, 2004). En revanche, ils se sont révélés beaucoup moins efficaces dans des systèmes impliquant la reconnaissance d'un très grand nombre d'activités parmi lesquelles certaines peuvent se

ressembler. De plus, les arbres de décision concèdent deux autres limites importantes qui sont, la nécessité d'avoir un très grand nombre de données pour être entraînés correctement et l'obligation de répéter la phase d'entraînement après chaque ajout d'une nouvelle activité ou après la modification des données d'une activité existante.

LE *CLUSTERING*

Les méthodes de *clustering* représentent une large partie des algorithmes d'apprentissage non-supervisés (Witten *et al.*, 2016). Le fonctionnement général de toutes les techniques de *clustering* existantes consiste en la division homogène d'un ensemble de données en différents *clusters* qui doivent partager des caractéristiques communes. Les *clusters* sont donc des sous-ensembles des données d'apprentissage. La séparation des données en *clusters* est réalisée, à la fois par minimisation de la distance intra-classe ; c'est-à-dire, la distance entre tous les éléments d'un même *cluster*, ainsi que par la maximisation de la distance inter-classe, qui correspond à la distance entre tous les *clusters*. Ensuite, pour chaque cluster, une donnée représentative lui est attribuée. En fonction de l'algorithme qui est employé, cette dernière peut être déterminée différemment. Dans certains cas, une nouvelle valeur (le barycentre) est calculée à l'aide des données déjà présentes dans le *cluster*. Dans d'autres cas, c'est une donnée existante qui sera élue comme la plus représentative pour chaque *cluster*. Un exemple d'une méthode de *clustering* utilisant le calcul des barycentres est illustré en Figure 2.8.

L'algorithme de *clustering* le plus utilisé dans le domaine de la reconnaissance d'activités est l'algorithme des *K*-moyennes (Messing *et al.*, 2009; Kovashka et Grauman, 2010). Cet algorithme requiert que le nombre total de *clusters K* soit fixé à l'avance. Ainsi, l'algorithme d'apprentissage commence par initialiser les barycentres, où la valeur qui leur est attribuée peut varier en fonction du problème à traiter. En effet, ceux-ci peuvent être initialisés à nul, avec des valeurs purement aléatoires ou encore avec des valeurs prises parmi les données exis-

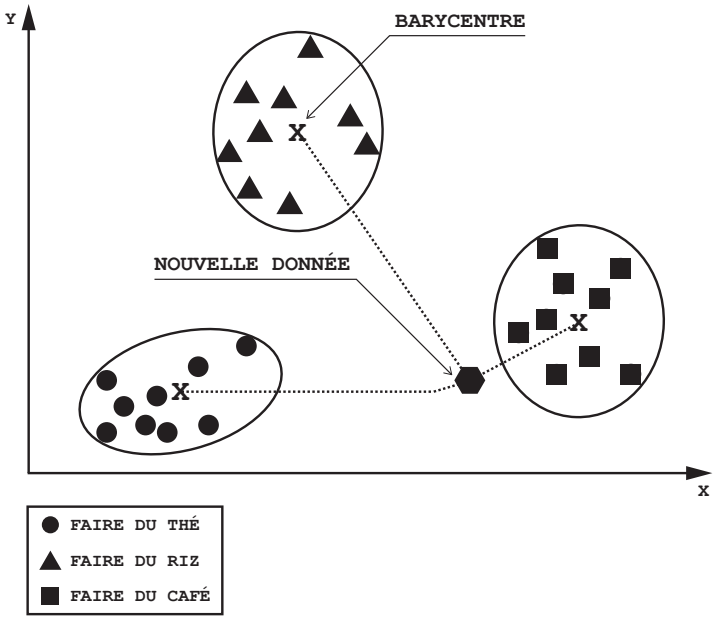


Figure 2.8 : Exemple d'un algorithme de *clustering*.

tantes. Lorsque les barycentres sont définis, l'algorithme va, de manière itérative, commencer par calculer la distance de chaque instance de l'ensemble de données par rapport à chacun des *K* barycentres. Chacune d'elles sera alors associée à son barycentre le plus proche. L'étape suivante consiste à mettre à jour la valeur des barycentres en fonction des *clusters* qui se sont formés. Enfin, ce processus est répété jusqu'à l'obtention d'une convergence, c'est-à-dire,

jusqu'à la stabilisation des valeurs des barycentres ; ce qui conduit à l'obtention du modèle d'apprentissage. Comme illustré dans l'exemple de la Figure 2.8, ce modèle d'apprentissage admet trois *clusters*, où chacun d'eux correspond au regroupement des données selon les trois mêmes activités qui sont utilisées depuis le début de ce chapitre. Dès lors, la classification d'une nouvelle instance nécessite de calculer la distance entre cette dernière et tous les barycentres. La plus petite distance obtenue détermine donc à quelle classe d'activité cette nouvelle donnée appartient. Dans cet exemple, la nouvelle donnée correspond donc à l'activité « *faire du café* ». Parmi toutes les mesures de distance qu'il est possible d'utiliser, celles qui sont les plus fréquemment utilisées dans l'algorithme des K-moyennes lorsque les données ont n dimensions sont, la distance euclidienne D_e et la distance de Manhattan D_m, respectivement rappelées par les Équations 2.13 et 2.14 tel que,

$$D_e = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{2.13}$$

$$D_m = \sum_{i=1}^{n} |x_i - y_i| \tag{2.14}$$

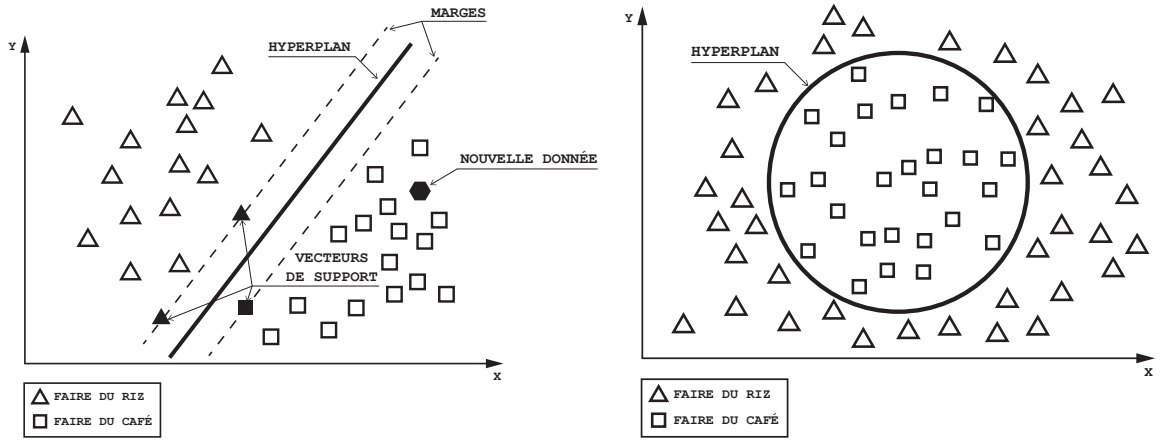
Bien que le côté non-supervisé de ces méthodes constitue un avantage majeur dans leur utilisation pour reconnaître des activités, elles n'en demeurent pas moins dénuées d'inconvénients. En effet, le processus d'entraînement est souvent coûteux en termes de temps d'exécution et en termes de consommation de la mémoire. De plus, le manque de flexibilité induit par la définition préalable du nombre de *clusters* est un inconvénient majeur pour un

système de reconnaissance où le nombre d'activités demeure potentiellement infini. Pour combler ce dernier inconvénient, il est possible de recourir à l'algorithme Density-Based Spatial Clustering of Applications with Noise (DBSCAN), car son utilisation permet de s'affranchir d'une initialisation manuelle du nombre de clusters (Gan et Tao, 2015). Néanmoins, celui-ci éprouve des difficultés lorsqu'il s'agit de gérer des clusters ayant des densités différentes et ceci est fréquent dans le domaine de la reconnaissance d'activités. DBSCAN n'est alors pas toujours le meilleur compromis.

LES MACHINES À VECTEURS DE SUPPORT

Les machines à support de vecteurs (Support Vector Machine (SVM)) font partie des techniques d'apprentissage supervisées. L'objectif d'un SVM est de créer une séparation des données d'entraînement optimale par le biais d'un hyperplan. De ce fait, l'algorithme va chercher à maximiser la marge, c'est-à-dire, la distance entre l'hyperplan et les données les plus proches de celui-ci. Ces données sont les supports de vecteurs. La Figure 2.9a montre un exemple de machine à support de vecteurs linéaire. En revanche, il est possible que dans certains cas, les données ne soient pas linéairement séparables, c'est-à-dire qu'aucun hyperplan de marge optimale n'existe. Pour remédier à cela, il est possible d'utiliser une fonction de noyau (linéaire, quadratique, polynomiale, de base radiale gaussienne, sigmoïde, *etc.*) qui permet alors de convertir des données d'apprentissage en un ensemble de dimension supérieur, où il est possible de créer un hyperplan de marge optimale permettant de séparer les données, tel qu'illustré en Figure 2.9b. Afin d'identifier à quelle classe d'activité la nouvelle donnée

appartient, le processus de reconnaissance va simplement calculer de quel côté de l'hyperplan celle-ci se situe. Ainsi, dans le cas de l'exemple proposé en Figure 2.9a la nouvelle donnée est à droite de l'hyperplan, elle est donc identifiée comme l'activité « *faire du café* ».



(a) Linéaire (b) Fonction de noyau

Figure 2.9 : Exemple d'un SVM linéaire ainsi que d'un SVM utilisant une fonction de noyau.

Les SVM sont souvent considérés comme des boîtes noires, car ils ne permettent pas l'extraction d'un modèle compréhensible, à l'inverse des arbres de décision, par exemple. Néanmoins, ils sont fréquemment utilisés dans le domaine de la reconnaissance d'activités pour plusieurs raisons (He et Jin, 2009; Anguita *et al.*, 2012). La première est qu'ils permettent d'obtenir des taux de reconnaissance très élevés. De plus, le processus de reconnaissance est très efficace en termes de consommation des ressources et peut donc facilement être réalisé sur des systèmes portables. Toutefois, les machines à support de vecteurs ne sont pas adaptées pour traiter un très gros volume de données lors du processus d'apprentissage—ce dernier impliquant de lourds calculs pour construire le modèle.

LES RÉSEAUX DE NEURONES ARTIFICIELS

Les réseaux neurones artificiels (Artificial Neural Networks ou ANNs) sont des méthodes d'apprentissage qui sont supervisées. Elles visent à imiter la pensée humaine par la modélisation simplifiée des systèmes neuronaux du cerveau de l'Homme et des animaux. Les ANNs sont composés de plusieurs neurones connectés entre eux qui s'échangent des signaux (Witten *et al.*, 2016). Chaque neurone est composé d'un nombre n d'entrées synaptiques qui sont chacune associées à un poids (w). Celles-ci sont ensuite agglomérées en une seule donnée grâce à un additionneur. Cette valeur est alors passée à une fonction d'activation qui va, pour chacun des neurones, renvoyer un signal de sortie positif ou négatif en fonction d'un certain seuil. Ainsi, pour effectuer un apprentissage supervisé avec les réseaux de neurones, la valeur des poids associés à chacune des synapses doit être modifiée afin que l'erreur entre les sorties du réseau et l'étiquette de la donnée testée soit atténuée. Dès que tous les poids sont mis à jour, la construction du modèle d'apprentissage est terminée et le processus reconnaissance peut commencer. La Figure 2.10 montre un exemple simplifié d'un perceptron multicouche, l'une des implémentations possibles des ANNs, permettant de reconnaître deux activités, « *faire du riz* » et « *faire du café* ». Dans cet exemple, les neurones d'entrée admettent les valeurs des capteurs de l'environnement intelligent et les neurones de sorties correspondent aux activités à reconnaître. Pour simplifier l'exemple, les valeurs de sorties sont exprimées sous la forme de booléens, mais les ANNs expriment normalement les sorties sous la forme de probabilités. Par conséquent, c'est la probabilité la plus élevée qui détermine l'étiquette de la donnée testée. Dans ce cas, lorsque le placard est ouvert ; que de l'eau s'écoule du robinet ; que la plaque de

cuisson avant-gauche est allumée ; que la casserole est sur la cuisinière et que la cafetière n'est pas allumée, l'activité prédite par cet ANN est « *faire du riz* ».

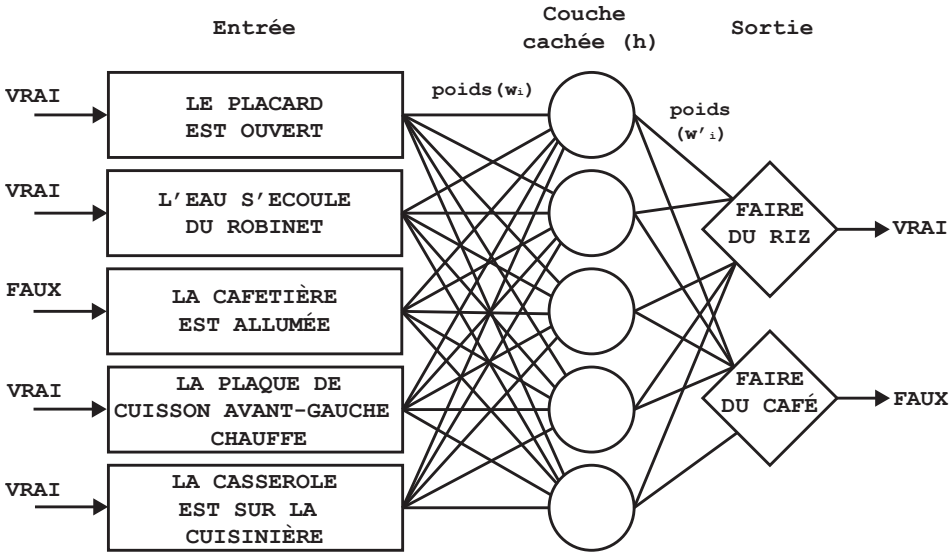


Figure 2.10 : Exemple de réseau de neurones artificiels.

De nombreux travaux dans le domaine de la reconnaissance d'activités ont démontré la performance des ANNs ainsi que leur robustesse quant à l'exploitation de données fortement bruitées (Pärkkä *et al.*, 2006; Delachaux *et al.*, 2013). Néanmoins, ces techniques nécessitent un temps d'apprentissage considérable pouvant atteindre plusieurs jours. Ceci demeure un inconvénient majeur d'autant plus que ce processus doit être reproduit après chaque ajout d'une nouvelle activité ou après la modification d'une activité déjà existante. De plus, il est très facile, pour un système de reconnaissance basé sur un réseau neuronal, de tomber dans le sur-apprentissage.

Dans la dernière décennie, l'augmentation exponentielle de la puissance des appareils informatiques a permis de propulser l'utilisation des techniques d'apprentissage profond (deep learning) qui sont des applications particulières des ANNs traditionnels qui ont été présenté précédemment. Celles-ci ont démontré leur excellente performance de reconnaissance dans beaucoup de domaines, dont celui de la reconnaissance d'activités (Yang *et al.*, 2015; Li *et al.*, 2016; Wang *et al.*, 2018). Bien qu'il soit possible que les techniques d'apprentissage profond soient capables de traiter des données brutes directement, c'est-à-dire de se passer de toutes les étapes préliminaires pour la construction du modèle d'apprentissage, elles admettent encore de nombreux inconvénients additionnels à ceux des ANNs. En effet, les modèles sont souvent utilisés comme des boîtes noires, car lorsque les résultats obtenus ne sont pas satisfaisants, il est pratiquement impossible d'en expliquer les raisons et surtout de trouver des solutions pour remédier au problème. De plus, ces techniques requièrent un volume conséquent de données pour être entraînées correctement et ainsi, obtenir un modèle générique et réutilisable.

2.2.3 LA MESURE DE LA PERFORMANCE

L'étape finale du processus d'apprentissage pour la reconnaissance d'activités consiste en une évaluation de la performance du système. Pour ce faire, il existe de nombreuses métriques qui s'appuient sur l'utilisation d'une matrice de confusion (Fawcett, 2006). Cette dernière permet l'identification de la relation qui lie la classe actuelle d'un enregistrement et celle qui est prédite par l'algorithme d'apprentissage. Par exemple, la matrice de confusion retournée par l'un de ces algorithmes dans le cas d'une reconnaissance binaire est donnée

par le Tableau 2.1. Lorsqu'une instance est positive et qu'elle est prédite comme positive, alors il s'agit d'un vrai positif (VP), sinon c'est un faux négatif (FN). Dans le cas contraire, si l'instance est négative, et qu'elle est prédite comme positive, alors s'agit d'un vrai négatif (VN), sinon c'est un faux positif (FP).

Tableau 2.1 : Matrice de confusion d'un système de reconnaissance binaire.

		classe prédite	
		oui	**non**
classe actuelle	**oui**	VP	FN
	non	FP	VN

Parmi l'ensemble des métriques permettant d'évaluer la performance d'un algorithme d'apprentissage, la mesure de la justesse est la plus fréquemment utilisée, du fait de sa simplicité. Elle permet de déterminer le ratio entre le nombre de prédictions correctement réalisées et le nombre total de cas tel que,

$$justesse = \frac{VP+VN}{VP+FP+VN+FN} \tag{2.15}$$

Néanmoins, malgré sa popularité, la justesse ne permet pas d'évaluer avec robustesse la qualité des prédictions. En effet, cette dernière ne prend pas en considération les prédictions qui peuvent être faites par chance. Par conséquent, une bonne justesse n'implique pas nécessairement une bonne performance de reconnaissance. C'est le phénomène du *paradoxe de la*

justesse. Pour tenir compte de cet effet de bord souvent négligé, d'autres métriques comme la Kappa de Cohen (κ) ou la *F-mesure* doivent être calculées (Ben-David, 2007). La première est exprimée par :

$$\kappa = \frac{P_o - P_e}{1 - P_e} \tag{2.16}$$

où P_o et P_e sont respectivement, les probabilités observées, c'est-à-dire le taux de succès obtenu par l'algorithme ; et les probabilités espérées, c'est-à-dire le taux de succès hypothétique de l'algorithme. L'Équation (2.17) permet, quant à elle, d'obtenir la *F-mesure*, telle que,

$$F\text{-}mesure = 2 \cdot \frac{précision \cdot rappel}{précision + rappel} \tag{2.17}$$

où la *précision* et le *rappel* sont respectivement donnés par les Équations 2.18 et 2.19 soit :

$$précision = \frac{VP}{VP + FP} \tag{2.18}$$

$$rappel = \frac{VP}{VP + FN} \tag{2.19}$$

2.3 LES *WORKBENCH* D'APPRENTISSAGE MACHINE

Selon Langlois et Lu (2008), un *workbench* d'apprentissage machine est un outil qui fourni une interface unifiée qui permet d'exploiter un certain nombre d'algorithmes d'apprentissage dans le but de traiter différents problèmes que ces méthodes peuvent potentiellement résoudre. Ainsi, plusieurs de ces outils ont été développés et exploités dans divers domaines de recherche tels que la bio-informatique (Larrañaga *et al.*, 2006), la cybersécurité (Handa *et al.*, 2019), la santé (Rajkomar *et al.*, 2019), et plus particulièrement pour la reconnaissance d'activités à l'intérieur des habitats intelligents (Chapron *et al.*, 2018; Ramirez-Prado *et al.*, 2019). Au sein des habitats intelligents, ces outils ont principalement permis de proposer un prototypage rapide des méthodes pour la reconnaissance d'activités ainsi qu'une meilleure réutilisation des composants logiciels permettant de reproduire les protocoles expérimentaux (Langlois et Lu, 2008). Par conséquent, puisque les *workbench* d'apprentissage machine, lorsqu'ils sont utilisés pour la reconnaissance d'activités au sein des habitats intelligents, constituent l'intermédiaire entre le matériel et l'exploitation logicielle, il semble important d'identifier le rôle et le fonctionnement de ses outils dans l'optique de mieux intégrer les *wearable devices* à ces environnements. Pour ce faire, cette section présente donc les trois principaux *workbench* d'apprentissage machine open-source qui sont parmi les plus utilisés dans la littérature, soit WEKA (Holmes *et al.*, 1994), RapidMiner (Ritthoo *et al.*, 2003; Hofmann et Klinkenberg, 2014) et Orange (Demšar *et al.*, 2004, 2013).

Par ailleurs, avec la tendance actuelle autour de l'intelligence artificielle, ce type d'outil devient de plus en plus populaire depuis que des fournisseurs de *cloud* publics, tels qu'*Amazon*

AWS, *Google Cloud Plateform* et *Microsoft Azure* ont introduit des versions grand public de ces applications. Comme elles font partie de la vaste liste de services payants proposés par chaque fournisseur, les *workbench* d'apprentissage machine peuvent bénéficier de l'évolutivité qu'offre ces nouvelles solutions. Cependant, comme ils sont considérés comme des services d'usage général, les mécanismes internes des techniques d'apprentissage machine ont été complètement occultés afin de les rendre accessibles à tous. Cette thèse ne prend donc pas en considération ce type d'outils, car ils ne représentent pas des solutions pertinentes dans un contexte de recherche académique.

2.3.1 WEKA

WEKA constitue un ensemble d'algorithmes d'apprentissage machine développé en Java qui a été introduit en 1994 principalement pour faciliter les tâches de forage de données (*data mining*) (Holmes *et al.*, 1994). Plus précisément, cet outil contient plusieurs fonctionnalités qui permettent d'effectuer du prétraitement de données, de la classification, de la régression, du *clustering*, des règles d'association, de l'apprentissage profond et de la visualisation. Bien que WEKA suggère son propre format de fichier natif (Attribute-Relation File Format ou ARFF), il supporte également les fichiers CSV, les fichiers Matlab ainsi que la connectivité à plusieurs bases de données *via* Java Database Connectivity (JDBC). En ce qui concerne le prétraitement des données, WEKA propose un grand nombre de méthodes allant de la simple suppression d'attributs pour les ensembles de données à des opérations avancées comme l'analyse par composantes principales (PCA).

Dans sa première version, WEKA était uniquement exploitable au travers d'une interface en ligne de commande (CLI). Aujourd'hui, bien qu'il soit toujours possible d'utiliser la CLI, cet outil offre la possibilité d'être inclus en tant que dépendance d'un projet Java ou R (Hornik *et al.*, 2009). De plus, il peut surtout être utilisé à travers plusieurs types d'interfaces graphiques. Dans un premier temps, la première possibilité demeure l'*explorer*, où l'utilisateur peut rapidement mettre en place des techniques de manipulation des données (filtrage, classification, *clustering* et visualisation), sans avoir besoin d'écrire de code. Dans un second temps, WEKA propose une interface propre aux expérimentations : l'*experimenter*. Il s'agit d'un outil permettant d'exécuter en parallèle, c'est-à-dire, selon différents processus sur une même machine ou sur différents ordinateurs d'un réseau, plusieurs expériences d'apprentissage machine afin d'évaluer les méthodes de classification et de régression. La troisième interface proposée par WEKA, appelée *knowledge flow*, offre la possibilité de réaliser les mêmes tâches qu'avec l'interface *explorer*. Néanmoins, celles-ci sont représentées sous forme de blocs qui doivent être manipulés pour construire un processus de flux opérationnel (*workflow*) comme le montre la Figure 2.11. Cette dernière illustre un exemple d'utilisation de l'interface *knowledge flow* pour un processus de classification sur les données du jeu de données IRIS (Asuncion et Newman, 2007). Dans cet exemple, l'algorithme k-NN, configuré avec $k = 1$ voisin a été utilisé et la performance de la classification a été évaluée selon la technique de la séparation en $80/20$, c'est-à-dire, 80% des données sont utilisées pour la phase d'entraînement, tandis que les 20% restants sont exploités lors de la phase de reconnaissance.

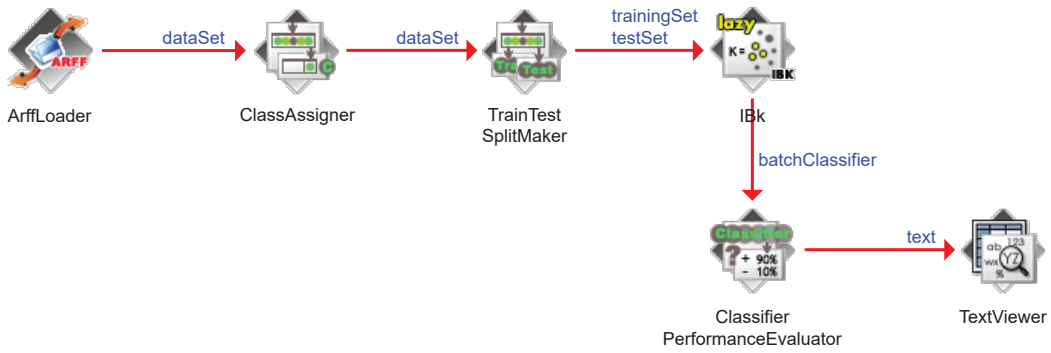


Figure 2.11 : Exemple de l'interface *knowledge flow* de WEKA qui décrit un processus de classification sur le jeu de données iris avec l'algorithme k-NN où $k = 1$ voisin et évaluée selon la technique de la séparation $80/20$.

WEKA est probablement le *workbench* d'apprentissage machine le plus connu et le plus utilisé puisqu'il s'agit d'un outil portable (Bouckaert *et al.*, 2010) offrant un large éventail de possibilités pour réaliser facilement des processus liés à l'apprentissage machine. Cependant, cet outil présente plusieurs inconvénients. Premièrement, comme il est écrit en Java et qu'il repose sur une ancienne base de code, sa capacité à traiter une grande quantité de données et la rapidité des différentes opérations sont profondément affectées par une consommation importante de ressources. Plus récemment, un gestionnaire de paquets a été introduit afin de permettre à des développeurs tiers d'étendre les fonctionnalités de base du *workbench* (Hall *et al.*, 2009). Néanmoins, bien que ces nouvelles fonctionnalités ont permis à WEKA de devenir plus flexible, le développement de ces paquets n'est possible qu'en Java et ceux-ci doivent respecter les contraintes d'intégration dans l'outil.

2.3.2 RAPIDMINER

RapidMiner, également connu sous le nom de YALE (Yet Another Learning Environment) (Ritthoo *et al.*, 2003; Hofmann et Klinkenberg, 2014), a été développé à partir de 2001 puis rebaptisé RapidMiner en 2007. De la même manière que WEKA, RapidMiner propose un environnement qui repose sur du code Java pour mettre en œuvre des techniques d'apprentissage machine. Cependant, contrairement à WEKA, cet outil n'offre qu'une seule interface qui permet de définir des flux opérationnels à la façon de l'interface *knowledge flow*. En effet, ces processus sont décrits à partir de blocs élémentaires appelés opérateurs. Il est alors possible pour l'utilisateur de composer une chaîne d'opérateurs en plaçant ces blocs sur un canevas et en câblant leurs ports d'entrée et de sortie, comme le montre la Figure 2.12. Bien que WEKA soit intégré dans RapidMiner en tant que dépendance utilisée pour plusieurs blocs, la majorité d'entre eux permettent d'étendre de nombreux aspects de l'apprentissage machine qui ne sont pas nécessairement couverts par WEKA.

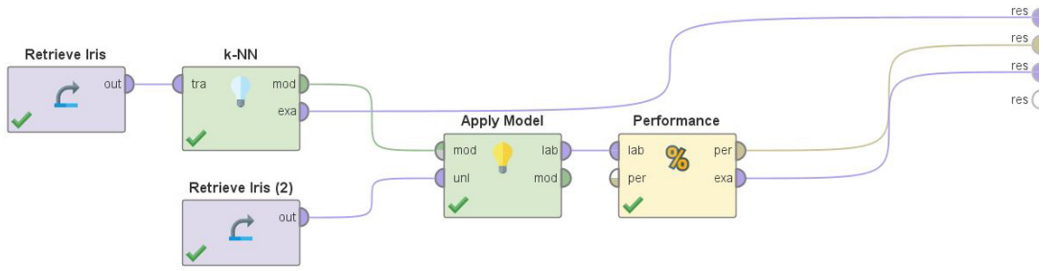


Figure 2.12 : Exemple de chaîne d'opérateurs dans l'outil RapidMiner qui décrit un processus de classification sur le jeu de données iris avec l'algorithme k-NN où $k = 1$ voisin et évaluée selon la technique de la séparation $80/20$.

Par ailleurs, RapidMiner inclut également la possibilité d'ajouter des paquets développés par des tiers afin d'améliorer ses capacités initiales. De plus, il permet l'utilisation d'un bloc de script où il est possible d'écrire du code en syntaxe Java ou Groovy. En ce sens, RapidMiner apparait tout aussi flexible que WEKA. Cependant, puisqu'il repose sur une conception logicielle comparable et un langage de programmation de base identique, ce *workbench* souffre des mêmes inconvénients. En outre, il est possible de dire que RapidMiner ne convient pas au déploiement d'applications expérimentales dans un contexte de recherche, car il ne dispose pas d'interface en ligne de commande et il n'est pas possible de l'utiliser comme une librairie dans une application.

2.3.3 ORANGE

Orange est le dernier *workbench* d'apprentissage machine, parmi les plus mentionnés dans la littérature, qui est présenté dans ce chapitre. Introduit en 1997, les fonctionnalités de base de cet outil ont été initialement conçues en `C++` et sa couche logicielle supérieure (c'est-à-dire les modules et l'interface graphique) était développée en Python (Demšar *et al.*, 2004, 2013). Cependant, depuis 2015, Orange a été entièrement redéveloppé. L'ancien noyau `C++` a été complètement remplacé par du code Python et l'interface graphique a également été revue. De la même manière que WEKA, Orange offre la possibilité soit d'être importé en tant que librairie dans un script Python, soit d'être utilisé comme outil à part entière grâce à interface graphique également basée sur l'utilisation de composants. En effet, comme le montre la Figure 2.13, Orange permet de manipuler des *widgets* disposés sur un canevas qui, une fois reliés entre eux, permettent de définir un *schéma* qui représente le flux opérationnel du processus d'apprentissage machine à réaliser. En outre, ce *workbench* offre également la possibilité d'importer des *widgets* personnalisés par l'intermédiaire d'un gestionnaire de paquets.

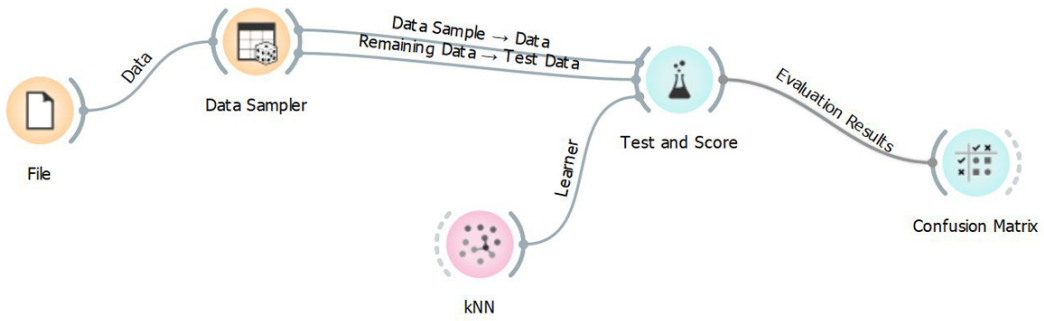


Figure 2.13 : Exemple d'un *schéma* dans Orange qui décrit un processus de classification sur le jeu de données iris avec l'algorithme k-NN où $k = 1$ voisin et évaluée selon la technique de la séparation $80/20$.

De la même manière que WEKA, puisqu'une intégration dans un programme Python est possible, Orange semble être tout autant susceptible d'être employé dans la conception d'applications académiques. De plus, étant donné que la version la plus récente d'Orange (Orange3) repose sur plusieurs outils modernes tels que NumPy [6], SciPy [7] et scikit-learn [8], les possibilités de créer des widgets personnalisés sont illimitées. Par ailleurs, l'exécution d'un processus de classification réalisé avec Orange a montré une consommation de ressources (temps de calcul et utilisation de la mémoire) plus raisonnable que pour le même processus effectué avec WEKA. En revanche, les recommandations fournies qui documentent la conception de widgets personnalisés ne sont pas triviales et se limitent à du code Python.

6. https://numpy.org
7. https://www.scipy.org
8. https://scikit-learn.org

2.4 CONCLUSION

Dans un premier temps, ce chapitre s'est intéressé aux habitats intelligents existants qui ont été regroupés en quatre catégories. Premièrement, les habitats intelligents académiques LIARA et le laboratoire DOMUS sont deux architectures qui reposent sur des technologies héritées du milieu industriel. Ces deux habitats, dont le coût total est très élevé, ont démontré une faiblesse en termes d'évolutivité et, par conséquent, l'absence d'une quelconque solution pour y intégrer facilement des *wearable devices*. Ensuite, les architectures basées composants comme Gator-Tech ou Amiqual4home ont démontré une meilleure capacité à intégrer matériellement de nouveaux capteurs, qu'ils soient de type ambiant ou portable. Cependant, bien que ce type d'architecture demeure beaucoup plus flexible, l'exploitation d'un environnement tel qu'OSGi reste relativement complexe et ne semble pas adaptée pour les *wearable devices*. Par ailleurs, il a également été constaté que ces habitats admettent des problématiques de fiabilité en raison de la surcharge du serveur central pouvant être occasionnée par un grand nombre de flux de données. Ceci ne fait donc pas des architectures qui s'appuient sur OSGi une solution idéale. D'autre part, les architectures qui reposent sur un réseau maillé ZigBee sont apparues comme les plus avancées en termes de flexibilité pour y intégrer facilement de nouveaux capteurs ambiants ou portables. Néanmoins, puisque très peu de dispositifs supportent nativement une telle technologie de communication, la possibilité d'y intégrer matériellement des *wearable devices* semble particulièrement limitée. Enfin, l'architecture distribuée qui repose sur la mise en œuvre de transducteurs intelligents introduite par Plantevin *et al.* (2018) constitue la meilleure approche en ce qui concerne la problématique d'intégration

matérielle de tous types de capteurs et plus particulièrement des *wearable devices*. De plus, cette architecture a démontré une excellente capacité à gérer de potentielles pannes sans altérer le fonctionnement du système complet. Ceci lui permet donc de garantir un excellent niveau de fiabilité. Toutefois, la conception de cette architecture peut encore être améliorée puisqu'elle fait intervenir une unité centrale. Bien que cette entité ne constitue pas un élément critique pour la fiabilité du système lors de la réalisation de la reconnaissance d'activités, elle pourrait aisément être remplacée par un système hautement disponible ce qui permettrait alors de supprimer complètement tout point de défaillance unique et le niveau de fiabilité du système s'en retrouverait donc encore plus élevé. Par ailleurs, si cette architecture facilite l'intégration matérielle de nouveaux capteurs, la prise en charge des composants logiciels, et plus spécifiquement ceux qui sont relatifs aux *wearable devices*, semble avoir été mise de côté dans la conception de cette architecture. Il apparait donc indispensable de proposer des solutions pour répondre à cette problématique.

Dans sa deuxième partie, ce chapitre a présenté plus en détails l'ensemble des étapes nécessaires pour accomplir le processus d'apprentissage pour reconnaître des activités. Les différentes techniques permettant de raffiner la qualité des données brutes renvoyées par les différemment capteurs et qui constituent la première phase du processus ont été exposés. Ensuite, les algorithmes de d'apprentissage les plus utilisés dans le domaine de la reconnaissance d'activités ont été examinés. Finalement, ce chapitre s'est achevé par la description des différentes méthodes permettant d'évaluer la performance de ces algorithmes et de la qualité intrinsèque de la reconnaissance d'activités.

Enfin, la dernière partie de ce chapitre s'est intéressé à trois principaux *workbench* d'apprentissage machine qui sont les plus employés dans la littérature relative à la reconnaissance d'activités dans les habitats intelligents. Puisque ceux-ci permettent de simplifier l'utilisation de nombreuses méthodes qui forment le processus d'apprentissage pour la reconnaissance d'activités, ces outils constituent l'intermédiaire entre les composants matériels présents dans les différentes architectures et l'exploitation logicielle qu'il en est fait. Ainsi, il a été montré que tous ces outils ont été développés selon des principes identiques, où l'ajout de nouvelles fonctionnalités est rendu possible soit par la création de paquets personnalisés, soit grâce à des modules génériques qui autorisent l'exécution de code. De ce fait, tous présentent le même principal inconvénient, c'est-à-dire une forte dépendance à leur contexte d'utilisation. Plus précisément, ces paquets additionnels doivent être conçus pour répondre aux exigences de l'outil pour lequel ils sont créés. Par conséquent, ce manque de flexibilité provoque indubitablement des problèmes d'intégration des *wearable devices* où la prise en charge de l'hétérogénéité des composants logiciels de ces dispositifs n'est pas supportée.

CHAPITRE III

LES *WEARABLE DEVICES* AU SEIN DES HABITATS INTELLIGENTS

Dans le chapitre précédent, le processus d'apprentissage pour la reconnaissance d'activités a été présenté dans un cas d'application générique. De nombreuses étapes, depuis l'obtention des données brutes jusqu'à la classification des activités à reconnaître, y ont été détaillées. Néanmoins, dans un contexte d'utilisation avec des *wearable devices*, la manière dont ces données sont acquises n'a, quant à elle, pas été mentionnée. De plus, l'adaptation de ce processus avec ces dispositifs fait intervenir une étape supplémentaire de transmission des données. Ce chapitre commence par définir la notion de *wearable devices*, puis il traite plus en détail des composants matériels et logiciels qui sont plus spécifiques dans l'utilisation de ces dispositifs au sein des habitats intelligents.

3.1 DÉFINITIONS

Selon le rapport publié par l'International Telecommunication Union (2012), l'internet des objets (Internet of Things ou IoT) représente une « infrastructure mondiale pour la société de l'information, qui permet de disposer de services évolués en interconnectant des objets (physiques ou virtuels) grâce aux technologies de l'information et de la communication interopérables existantes ou en évolution ». Ainsi, d'un point de vue conceptuel, l'IoT caractérise des objets physiques connectés ayant leur propre identité numérique et la capacité de communiquer les uns avec les autres. Ce réseau crée, en quelque sorte, une passerelle entre le

monde physique et le monde virtuel. En d'autres termes, il s'agit de fournir une identification numérique directe et normalisée (adresse IP, protocoles de communication, *etc.*) d'un objet physique grâce à un système de communication sans-fil qui peut être du Bluetooth ou encore du Wi-Fi.

Par conséquent, les *wearable devices* appartiennent aux objets connectés, puisqu'ils se caractérisent plus particulièrement comme une technologie qui embarque des capteurs et qui est disposée directement sur le corps humain. Godfrey *et al.* (2018) ont identifié deux types de *wearable devices* : les dispositifs autonomes (*p. ex.* les moniteurs d'activités physiques) et les dispositifs de mesure (*p. ex.* un moniteur de fréquence cardiaque porté sur la poitrine). Les dispositifs autonomes permettent de réaliser différents traitements, plus ou moins complexes, dont les résultats sont ensuite communiqués à d'autres appareils, tandis que les dispositifs de mesure ont, quant à eux, pour seul objectif de transférer les informations du capteur vers un serveur, par exemple.

Les principaux avantages offerts par les *wearable devices*, comme leur taille, leur faible prix ou leur facilité d'utilisation, ont favorisé leur usage dans différents domaines de recherche tels que la surveillance des activités physiques et sportives, la surveillance en continu de la santé ou encore la reconnaissance de gestes, d'activités ou de chutes. (Seon-Woo Lee et Mase, 2002; Istepanian *et al.*, 2011; Garcia-Ceja *et al.*, 2014; Bayat *et al.*, 2014; Yuan Jie Fan *et al.*, 2014; Gao *et al.*, 2014; Nielsen, 2014; Adib *et al.*, 2015; Davis *et al.*, 2016; Khan *et al.*, 2016; Chapron *et al.*, 2018).

3.2 LES CAPTEURS

Pour réaliser le processus d'apprentissage permettant de mettre en place une reconnaissance, la première étape consiste à récolter des données brutes qui sont produites par des capteurs. Cependant, puisqu'il en existe une grande diversité, il semble nécessaire d'identifier ceux qui sont les plus adéquats pour être proposés en tant que *wearable devices*. Pour ce faire, cette section présente, en ce basant sur la taxonomie d'Acampora *et al.* (2013), les capteurs parmi ceux qui sont les plus utilisés dans ce domaine d'application.

3.2.1 LES CAPTEURS DE MOUVEMENT

Les centrales inertielles (Inertial Measurement Units ou IMUs) sont les capteurs de mouvement qui sont le plus fréquemment implantés dans les *wearable devices* de par leur simplicité d'utilisation. Celles-ci peuvent comporter d'un à trois types de capteurs différents : les accéléromètres, les gyroscopes et les magnétomètres. Ils permettent de mesurer respectivement l'accélération linéaire, la vitesse angulaire et l'intensité du champ magnétique que subissent les objets auxquels ils sont fixés, grâce à un maximum de trois axes orthogonaux (x, y et z). L'adoption de ces capteurs a permis de nombreuses applications concrètes, telles que la reconnaissance de chutes chez les personnes âgées, l'identification de l'orientation du corps, la réadaptation, ainsi que la reconnaissance d'activités quotidiennes (Seon-Woo Lee et Mase, 2002; Garcia-Ceja *et al.*, 2014; Bayat *et al.*, 2014; Gao *et al.*, 2014; Davis *et al.*, 2016; Chapron *et al.*, 2018).

3.2.2 LES CAPTEURS PHYSIOLOGIQUES

Grâce aux progrès technologiques et plus particulièrement à la miniaturisation de l'électronique, de nouveaux types de capteurs, jusqu'alors réservés au domaine médical, ont pu être utilisés pour récolter les données physiologiques des utilisateurs, sans pour autant recourir aux services spécialisés d'un hôpital. Cependant, certaines techniques sont encore considérées comme complexes et intrusives à la fois. C'est, par exemple, le cas de l'électroencéphalogramme (EEG) qui demeure encore rarement exploité. Ce capteur physiologique permet d'obtenir la représentation de l'activité électrique du cerveau par l'intermédiaire d'électrodes disposées sur le crâne d'un individu. Néanmoins, plusieurs autres capteurs physiologiques ont, quant à eux, été très largement utilisés, et ce, dans de nombreux domaines de recherche. Parmi ceux-ci, il est possible de mentionner les capteurs qui permettent l'acquisition de l'électrocardiogramme (ECG), c'est-à-dire, la représentation graphique de l'activité électrique du cœur ; de l'électromyogramme (EMG), soit l'activité électrique produite par les muscles ; de capteurs mesurant la glycémie, qui indique le taux de concentration de glucose dans le sang, ou de la Saturation Pulsée en Oxygène (SPO_2), qui désigne le taux de saturation du sang en oxygène. Au sein des habitats intelligents, l'intégration de ces capteurs dans des *wearable devices* a principalement permis de proposer de nouvelles techniques pour la réhabilitation (Yuan Jie Fan *et al.*, 2014), la reconnaissance de gestes (Jung *et al.*, 2015; Benatti *et al.*, 2015; Tavakoli *et al.*, 2018) et la surveillance en continu de la santé des résidents pour détecter de possibles maladies (Istepanian *et al.*, 2011; Adib *et al.*, 2015; Khan *et al.*, 2016).

3.2.3 LES CAPTEURS DE COURBURE ET DE FORCE

Certains dispositifs considérés comme des *wearable devices* tels que les gants présentés par Sanford *et al.* (2015) et Zheng *et al.* (2016) ainsi que les chaussures conçues par Bamberg *et al.* (2008) et Bae et Tomizuka (2013), ont nécessité l'utilisation de capteurs de courbure (*flex sensors*) et de force (Force Sensitive Resistors ou FSRs). Ces capteurs, respectivement montrés en Figures 3.1a et 3.1b fonctionnent grâce à des résistances dont la valeur change respectivement en fonction de la courbe qui est donnée au capteur ou de la pression qui est appliquée sur la surface de contact. Ainsi, comme illustré par ces différentes recherches, ces capteurs se sont montrés utiles dans certaines applications de réadaptation et de reconnaissance de gestes et d'activités.

(a) Capteur de courbure (b) Capteur de force (FSR)

Figure 3.1 : Exemples de capteurs de courbure (a) et de force (b).

3.2.4 LES CAPTEURS ENVIRONNEMENTAUX

Les capteurs environnementaux comportent principalement les capteurs de pression barométrique, d'humidité, de température et de dioxyde de carbone (CO_2). Ils sont principalement utilisés pour fournir des informations relatives à l'environnement proche des utilisateurs de *wearable devices*. Ainsi, les données produites par ce type de capteur peuvent être utilisées pour identifier le contexte d'utilisation dans lequel leurs porteurs évoluent et par conséquent, renforcer les analyses réalisées grâce aux autres capteurs (Acampora *et al.*, 2013). Par ailleurs, certains capteurs d'humidité et de température ont également été utilisés pour fournir des informations nécessaires à la surveillance de l'état de santé des utilisateurs des *wearable devices* (Anliker *et al.*, 2004).

3.3 LES TECHNOLOGIES DE COMMUNICATION SANS-FIL

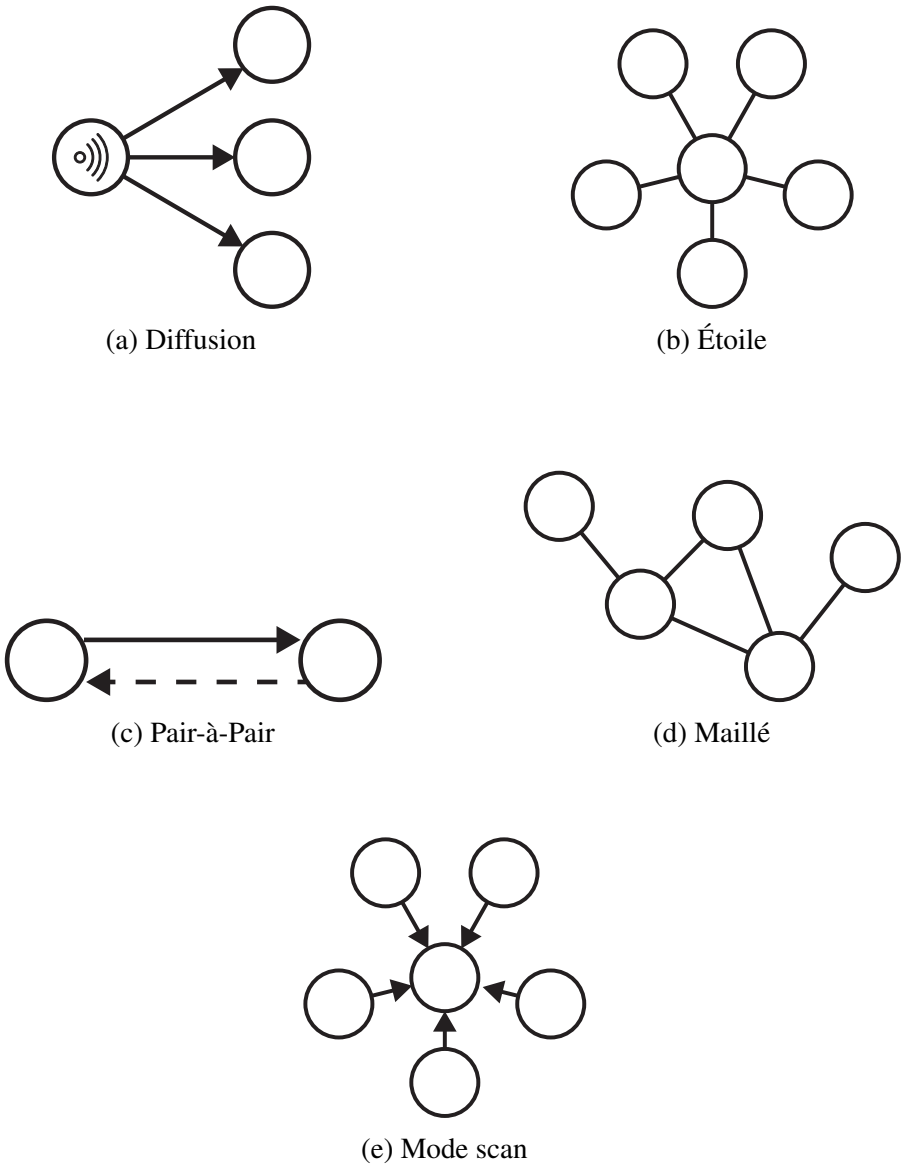
Lorsqu'il est appliqué à des *wearable devices*, le processus d'apprentissage nécessite une étape supplémentaire qui est la transmission de données. Ces dernières peuvent être les données brutes, les caractéristiques ou les décisions qui sont émises en sortie de l'algorithme d'apprentissage. En fonction des cas et des capteurs que peut comporter le dispositif, la fréquence d'émission ainsi que le poids des données à transmettre peut fortement varier. Ainsi, il est nécessaire de choisir une technologie de communication sans-fil adéquate pour chaque situation. Cette section va donc s'intéresser à celles qui sont les plus communément utilisées par les *wearable devices* existants.

3.3.1 LES TOPOLOGIES DES RÉSEAUX

Avant d'entrer dans le détail des technologies de communication utilisées par les *wearable devices*, il est important de commencer par énoncer les topologies de réseaux existantes dans le domaine du sans-fil. En effet, elles jouent un rôle important dans le bon fonctionnement de la communication réseau et l'identification de leur différentes caractéristiques va permettre, *a posteriori*, de mieux cibler les besoins pour le développement de nouveaux systèmes. D'après Tanenbaum et Wetherall (2010), les topologies les plus fréquemment mises en place dans un contexte sans-fil sont :

- La **diffusion (*broadcast*)** (Figure 3.2a) où un message est envoyé par un nœud émetteur à tous les nœuds récepteurs du réseau qui sont à sa portée. Le canal de communication est unidirectionnel et aucun accusé de réception n'est transmis au nœud émetteur depuis les nœuds récepteurs.
- Les **réseaux en étoile** (Figure 3.2b) admettent un nœud émetteur-récepteur central qui communique, à travers plusieurs canaux bidirectionnels, avec plusieurs autres nœuds émetteurs-récepteurs périphériques. Ces émetteurs-récepteurs périphériques ne peuvent pas communiquer directement les uns avec les autres.
- Les **réseaux pair-à-pair (Peer-to-Peer ou P2P)** permettent à deux nœuds émetteurs-récepteurs, reliés par un canal de communication bidirectionnel, d'échanger des données dans les deux sens, tel qu'illustré par la Figure 3.2c.

- Les **réseaux maillés** (Figure 3.2d) permettent l'échange de données entre n'importe quels nœuds du réseau sans passer par un nœud central. La communication est bidirectionnelle et chaque nœud peut-être relié à plusieurs autres nœud qui composent le réseau.
- Le **mode scan** (Figure 3.2e) permet à un nœud central de fonctionner en mode réception, c'est-à-dire, en attente de recevoir un signal provenant de n'importe quel nœud émetteur qui est à sa portée. La communication entre les deux nœuds est unidirectionnelle et se fait du nœud émetteur vers le nœud central.



(a) Diffusion

(b) Étoile

(c) Pair-à-Pair

(d) Maillé

(e) Mode scan

Figure 3.2 : Topologies réseaux pour les technologies de communication sans-fil.

3.3.2 WI-FI

Le Wi-Fi, régi par le standard IEEE 802.11 [9], est probablement la technologie de communication sans-fil la plus connue, car nous l'utilisons massivement depuis deux décennies. En constante amélioration, chaque génération du standard a apporté aussi bien des vitesses plus rapides et une latence réduite, qu'une meilleure expérience utilisateur dans une multitude d'environnements et avec divers types de périphériques. Toutes utilisations confondues, le standard qui est actuellement le plus utilisé est le Wi-Fi 802.11ac. Bien qu'il offre des vitesses de transfert de données à haut débit (de 400 Mbit/s à 2.6 Gbit/s), sa portée reste dans la moyenne de celles proposées par d'autres technologies, soit plusieurs dizaines de mètres. Néanmoins, dans le contexte des *werable devices*, le standard 802.11n sur la bande de Fréquence 2.4GHz, qui est le plus utilisé. Bien que celui-ci permette d'obtenir une portée plus élevée de quelques dizaines de mètres, les vitesses de transfert de données demeurent plus faibles (de 72 à 288 Mbit/s). Par ailleurs, le plus gros inconvénient de cette technologie reste sa forte consommation énergétique, ce qui est un facteur extrêmement limitant pour les objets connectés (IoT), parmi lesquels se retrouvent les *wearable devices*.

Dans un futur proche, les évolutions prévues dans les standards 802.11ah et 802.11ax vont principalement se concentrer sur l'amélioration de la consommation énergétique afin que l'utilisation du Wi-Fi devienne adaptée pour l'IoT. Néanmoins, la contrepartie de cette

9. https://goo.gl/X9WKyx

évolution est l'impact sur la vitesse de transfert de données, qui va alors, devenir beaucoup plus faible (8 Mbit/s) (Sun *et al.*, 2013).

3.3.3 BLUETOOTH LOW ENERGY (BLE)

Le BLE [10] a été lancé en 2010 dans le cadre de la spécification du Bluetooth 4.0. Bien souvent, il est considéré comme une version plus légère et plus optimisée du Bluetooth classique (versions 1 à 3), mais en réalité, le BLE présente une conception totalement différente. En effet, ce dernier a été pensé pour être une technologie de communication proposant une consommation d'énergie très faible, spécifiquement optimisée pour pouvoir offrir un coût réduit, une faible bande passante, ainsi qu'une complexité moindre. En comparaison aux autres technologies de communication sans-fil à faible consommation, le BLE a connu une adoption très rapide principalement grâce à la croissance phénoménale des téléphones intelligents et plus généralement de l'informatique mobile. Ainsi, les leaders de l'industrie mobile comme Apple ou Samsung ont favorisé le large déploiement de cette technologie qui est aujourd'hui, la technologie de communication la plus utilisée par les *wearable devices* (Gomez *et al.*, 2012).

D'autre part, avec l'arrivée des premiers appareils supportant la version 5 du Bluetooth, cette technologie devient la seule à supporter, intuitivement, l'intégralité des topologies réseaux présentées dans la Sous-Section 3.3.1. En effet, en plus d'offrir un débit et une portée deux fois supérieurs à celui de la quatrième version, soit 2 Mbit/s et 250 mètres respectivement,

10. https://www.bluetooth.com/specifications

cette nouvelle spécification intègre également la nouvelle norme : « Bluetooth Mesh 1.0 », qui va permettre la mise en place d'un réseau maillé [11]. Cependant, il ne sera pas possible de tirer profit de tous ces nouveaux avantages en même temps, puisque cette cinquième version permettra deux modes de fonctionnement : haut débit ou longue portée. Ainsi, dans le premier cas, la vitesse de transfert sera favorisée au profit de la portée et inversement, ceci dans le but de continuer les efforts pour la réduction de la consommation d'énergie tout en améliorant les capacités.

3.3.4 ZIGBEE

La technologie ZigBee a été développée dans les années 1990 et avec l'avènement des objets connectés, elle est devenue l'une des technologies de communication parmi les plus utilisées. En comparaison avec le Wi-Fi, le ZigBee admet l'avantage de ne consommer que peu d'énergie. De plus, le standard IEEE 802.15.4 [12], qui régit cette technologie, indique que le ZigBee peut offrir une porté de plusieurs kilomètres, en fonction de la puissance de l'émetteur et de certaines caractéristiques environnementales.

Initialement, le ZigBee a été introduit pour répondre à des problématiques liées à l'hétérogénéité de l'IoT (Rahmani *et al.*, 2015; Cho *et al.*, 2013) et par extension, des technologies présentes au sein des habitats intelligents (Hui *et al.*, 2017). ZigBee permet de s'appuyer sur la conception d'une topologie de réseau maillé (*mesh*). De ce fait, certains *wearable devices*

11. https://www.bluetooth.com/specifications/mesh-specifications

12. https://goo.gl/X9WKyx

(Cruz *et al.*, 2018) et habitats intelligents (Cook *et al.*, 2013) ont vite adopté cette technologie, ce qui a, par exemple, permis à Cook *et al.* (2013), d'unifier et de simplifier la communication entre tous les capteurs présents dans l'habitat CASAS.

3.3.5 LA COMMUNICATION EN CHAMP PROCHE

La communication en champ proche (Near Field Communication ou NFC) est une technologie de communication sans-fil à faible consommation et de courte portée, permettant l'échange d'informations entre des périphériques jusqu'à une distance de l'ordre du centimètre. L'avantage principal de cette technologie est que les périphériques NFC passifs, par exemple, les cartes de crédit, ne requièrent aucune alimentation. Ils deviennent actifs uniquement lorsqu'ils entrent dans le champ proche d'un périphérique qui est alimenté (*p. ex.* un lecteur).

Cette technologie de communication s'est principalement popularisée grâce aux méthodes de paiement sans contact (Ondrus et Pigneur, 2007), mais elle a également été utilisée dans plusieurs recherches relatives aux environnements intelligents. Par exemple, Pering *et al.* (2007) ont proposé un système de reconnaissance de gestes qui exploite le NFC d'un téléphone intelligent. Aussi, Chang *et al.* (2009) ont proposé une architecture pour l'automatisation du contrôle des installations électriques d'une maison, grâce à un système de reconnaissance d'utilisateurs de téléphones intelligents qui s'appuie principalement sur une détection d'appareils NFC présents dans l'environnement.

La limitation de cette technologie en termes de distance ne fait pas du NFC un concurrent direct du BLE ou du ZigBee. Son adoption concerne plutôt un marché de niche et il est préférable de le considérer comme complémentaire aux autres technologies de communication sans-fil présentées dans ce chapitre.

3.3.6 BILAN DES TECHNOLOGIES DE COMMUNICATION SANS-FIL

Dans cette section, les technologies de communications sans-fil les plus utilisées par les *wearable devices* ont été présentées. Puisqu'elles admettent des caractéristiques différentes, il convient donc d'en proposer une synthèse. Pour ce faire, le Tableau 3.1 illustre les différences techniques pertinentes pour ces technologies. La consommation énergétique est identifiée, pour chacune d'elles, par les valeurs théoriques de la consommation moyenne ainsi que du

Tableau 3.1 : Caractéristiques des différentes technologies de communication sans-fil employées par les *wearable devices*

	Portée	**Débit**	**Latence**	**Pic de consommation**	**Consommation moyenne**
Wi-Fi (812.11n)	70 m	150 - 600 MB/s	$\approx 1\,ms$	150 mA	100 mA
Wi-Fi (812.11ac)	35 m	433 - 2600 MB/s	$\approx 1\,ms$	150 mA	100 mA
BLE	250 m	1 - 2 MB/s	$\approx 1\,ms$	15 mA	25 μA
ZigBee	> 1 km	250 kB/s	$\approx 1\,s$	30 mA	30 mA
NFC	0.1 m	400 kB/s	$\approx 1\,s$	50 mA	50 mA

pic de consommation, c'est-à-dire, la valeur maximum de l'intensité du courant. De plus, la latence théorique y est également proposée. Puisque cette dernière caractéristique permet de mesurer le temps nécessaire à la transmission d'un signal entre un émetteur et son récepteur, elle aura donc un impact significatif sur la consommation énergétique. En effet, une forte latence va permettre de consommer moins d'énergie et inversement dans le cas d'une latence faible.

Dans le vaste monde des technologies sans-fil et plus particulièrement celles à faible consommation d'énergie, seules quelques-unes d'entres elles se sont popularisées avec le développement de l'IoT et des *wearable devices*. Parmi celles-ci, il est possible de retrouver le Wi-Fi, le BLE, le ZigBee et le NFC. Bien que chacune d'elles puisse être utilisée avec ces dispositifs ayant une autonomie de fonctionnement limitée, elles admettent des capacités de portée, de débit et de robustesse différentes. Ces variations de performances impliquent que chaque méthode de communication possède un cas d'application qui lui est propre. Le choix de la technologie à adopter dans le processus de conception de *wearable devices* est donc crucial et doit être effectué rigoureusement en fonction de l'utilisation qui doit en être faite.

3.4 LES ÉCHANGES DE DONNÉES

Dans la section précédente, les différentes technologies de communication sans-fil ont été présentées. Ces différentes technologies constituent la couche matérielle du modèle de communication réseau. Par conséquent, cette section s'intéresse plus en détail aux différents protocoles de communication appartenant, quant à eux, aux couches hautes de ce même

modèle de communication et qui sont les plus utilisés dans le contexte des réseaux de capteurs sans-fil.

3.4.1 LE MODÈLE *PUBLISH/SUBSCRIBE*

Depuis l'arrivée des habitats intelligents, certaines architectures comme celle proposée pour CASAS (Cook *et al.*, 2013) ont adopté le modèle *publish/subscribe* pour l'ensemble de leurs réseaux de capteurs. Depuis, plusieurs travaux se sont principalement intéressés à l'utilisation de ce modèle pour l'intégration de l'IoT au sein de ces habitats (Lee *et al.*, 2014; Upadhyay *et al.*, 2016; Van Den Bossche *et al.*, 2018).

Le modèle *publish/subscribe* est une alternative au modèle client/serveur traditionnel où un client communique directement avec le serveur. Dans se modèle, il est possible de distinguer deux types de clients : ceux qui envoient des messages, les *publishers* et ceux qui les reçoivent, les *subscribers*. Dans une vaste majorité des utilisations de ce modèle, les *publishers* et les *subscribers* n'échangent jamais directement et il ne savent pas que l'autre existe physiquement. Le lien entre eux est fait par l'intermédiaire d'un troisième composant, le *broker*. Son rôle est de filter les messages entrant et de les redistribuer correctement aux différents *subscribers* comme illustré par la Figure 3.3. L'avantage principal du modèle *publish/subscribe* est son élasticité. En effet, il est assez simple de voir comment il est possible de paralléliser les opérations exécutées par le *broker*. De plus, les messages peuvent être gérés comme des évènements qu'il ne restera alors qu'a intercepter.

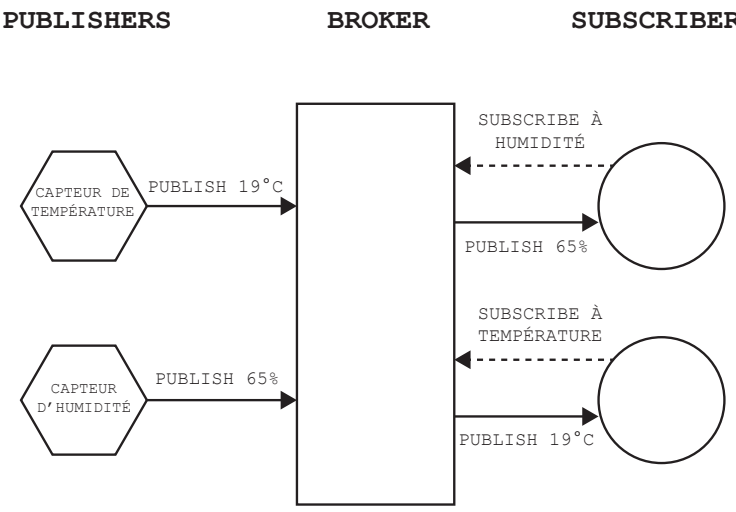


Figure 3.3 : Fonctionnement du modèle de communication de type *publish/subscribe*.

Le modèle *publish/subscribe* connaît de nombreuses implémentations parmi lesquelles il est possible de retrouver Advanced Message Queuing Protocol (AMQP) (Vinoski, 2006), XMPP (Saint-Andre, 2011), RabbitMQ (Dossot, 2014) et ZéroMQ (Hintjens, 2013). Cependant, Message Queuing Telemetry Transport (MQTT) est la plus connue d'entre elles (Hunkeler *et al.*, 2008). En effet, grâce à sa portabilité [13] et sa faible consommation de ressources (débit, mémoire et consommation d'énergie), elle demeure une excellente solution pour l'échange de données dans le domaine de l'IoT, où la puissance des dispositifs reste encore relativement limitée. En plus des avantages offerts par le modèle *publish/subscribe*, MQTT propose la définition de trois différents niveaux de qualité de service (Quality of Service ou QoS) : Le message est distribué une fois tout au plus, ou n'est pas distribué du tout ; le message est toujours distribué au moins une fois ou le message est toujours distribué une seule fois. Ceux-ci permettent donc une plus grande flexibilité en ce qui concerne le

13. https://github.com/mqtt/mqtt.github.io/wiki/libraries

niveau de fiabilité requis par le système ; c'est-à-dire, la garantie que les messages sont, ou non, correctement envoyés et reçus. Aussi, dans son objectif de demeurer une solution légère, MQTT permet d'avoir recours à une option de sécurité relativement simple. Cette dernière consiste en une authentification des clients auprès du *broker via* un nom d'utilisateur et un mot de passe. Néanmoins, plusieurs couches de sécurité supplémentaires peuvent y être ajoutées (utilisation d'un réseau privé virtuel (Virtual Private Network ou VPN) ou le chiffrement des échanges de messages par SSL/TLS, *etc.*)

3.4.2 LE CAS SPÉCIFIQUE DU BLE

D'après le rapport publié par ON World Inc. (2017), le BLE serait la technologie de communication sans-fil la plus utilisée dans le domaine de l'IoT et plus particulièrement des *wearable devices*. De ce fait, la Sous-Section 3.3.3 ayant présenté les principales caractéristiques techniques de bas niveau pour cette technologie, il est désormais nécessaire d'en présenter le fonctionnement de haut niveau. En effet, la communication par BLE oblige l'utilisation du protocole tel que défini par le standard. Par conséquent, l'implémentation d'un tout autre protocole de haut niveau déjà existant n'est pas possible avec cette technologie.

Le protocole du BLE est divisé en deux catégories : le contrôleur et l'hôte ; chacune admettant des sous-catégories parmi lesquelles il est possible de retrouver le Generic Access Profile (GAP) et le Generic ATTribute (GATT). Le GAP définit la topologie générale du réseau. En d'autres termes, si un dispositif Bluetooth est visible par d'autres, c'est par l'intermédiaire de ce profil. Il détermine comment les appareils peuvent, ou non, interagir entre eux. Le

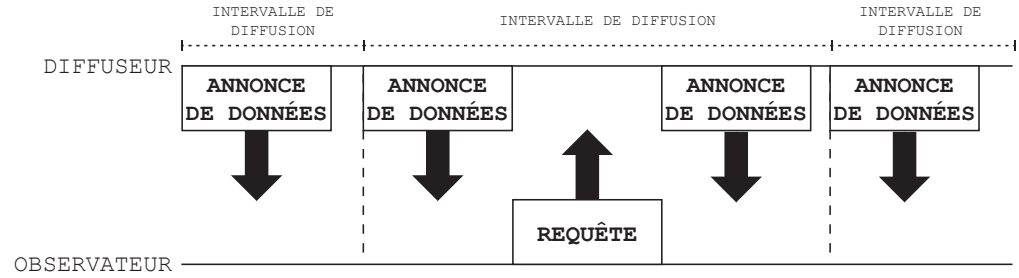
GATT, quant à lui, décrit en détail la manière dont les données sont formatées, conditionnées et transférées selon les règles définies par l'ATTribute Protocol (ATT). Pour communiquer avec le monde extérieur, un appareil BLE peut se trouver en deux modes différents : le mode diffusion (Figure 3.2a) ou le mode communication, qui correspond à une topologie étoile (Figure 3.2b). Ceux-ci sont définis dans les directives relatives au GAP.

Dans le cas du mode diffusion, il est important de distinguer deux rôles que peuvent avoir les appareils BLE : les diffuseurs et les observateurs. Pendant un intervalle de temps donné (l'intervalle de diffusion), le diffuseur est responsable d'annoncer publiquement des données. Si un observateur demande à récupérer les données annoncées par le diffuseur, ce dernier doit alors lui transmettre—sinon elles sont annoncées de nouveau dès lors que l'intervalle de diffusion est écoulé. Un nouveau cycle peut alors recommencer. En outre, il est important de noter qu'aucune connexion n'est établie entre un observateur et un diffuseur. Le fonctionnement de ce processus est illustré par la Figure 3.4a.

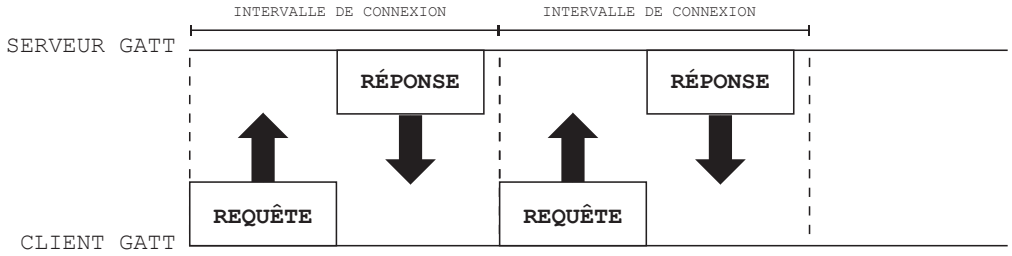
À l'inverse du mode diffusion, deux appareils BLE, lorsqu'ils sont en mode communication, doivent explicitement établir une connexion entre eux pour que des données puissent être échangées. Si un observateur demande une connexion à un diffuseur, le processus de diffusion s'arrête et il n'est plus possible d'obtenir les données qui étaient annoncées. Ce dernier adopte alors le rôle de périphérique et l'observateur devient le central. Un appareil BLE ayant le rôle de périphérique ne peut se connecter qu'à un seul dispositif central à la fois, mais un central peut, quant à lui, être connecté à plusieurs périphériques. Cependant, la connexion peut être

interrompue intentionnellement ou non (*p. ex.* perte de l'alimentation), et ce, par n'importe quel dispositif.

Dès que la connexion est établie, la communication entre le dispositif central et le dispositif périphérique peut commencer. Le dispositif qui demande les données joue alors le rôle de serveur GATT. Puisque les rôles définis par le GAP et le GATT sont indépendants, le central et le périphérique peuvent tous deux devenir le serveur. Celui-ci est donc sollicité par le client GATT qui lui envoie des requêtes. Le serveur GATT indique un intervalle de connexion au client qui va alors essayer de se reconnecter après chaque délai pour récupérer de nouvelles données, si elles existent. Une représentation graphique du fonctionnement de ce processus est donnée en Figure 3.4b. Les données sont transmises du serveur au client par une collection de services. Ceux-ci sont utilisés pour diviser les données en entités logiques et contiennent des blocs de données spécifiques qui sont les caractéristiques. Un service peut avoir une ou plusieurs caractéristiques et chaque service se distingue des autres au moyen d'un identifiant numérique unique (Universally Unique IDentifier ou UUID), qui peut être défini soit sur 16 bits (pour les services BLE officiels), soit sur 128 bits (pour les services personnalisés). Les caractéristiques sont elles aussi identifiées par un UUID de 16 ou de 128 bits prédéfini. C'est *via* celles-ci que les informations sont échangées, car contrairement aux services, elles ne peuvent encapsuler qu'une unique donnée (*p. ex.* une valeur binaire, un entier, un tableau de valeurs).



(a) Échange de données en mode diffusion



(b) Échange de données en mode connexion

Figure 3.4 : Processus d'échange de données dans les deux modes de fonctionnement du BLE, soit les modes diffusion et connexion.

3.4.3 LES SERVICES WEB

Les services web permettent à différentes applications de communiquer entre elles en fournissant une plateforme commune pour l'échange de données. Les requêtes et les réponses émises par les applications sont soumises à des standards. Parmi les plus populaires, il est possible de retrouver le protocole Simple Object Access Protocol (SOAP), l'architecture Representational State Transfer (REST) et le protocole Constrained Application Protocol (CoAP).

LE PROTOCOLE SOAP

Le protocole SOAP, tout comme le modèle *publish/subscribe*, est lui aussi utilisé dans les habitats intelligents depuis leur apparition. En effet, c'est principalement le cas de ceux ayant opté pour une architecture par composants et plus particulièrement OSGi (comme Gator-tech (Helal *et al.*, 2005)), puisque cette technologie repose sur SOAP pour tirer profit des services web. Depuis, de nombreux travaux se sont intéressés à l'interopérabilité des données ambiantes fournies par les habitats intelligents avec les données produites par les *wearable devices* (Perumal *et al.*, 2008; Cubo *et al.*, 2014; Díaz-Rodríguez *et al.*, 2018).

SOAP est un protocole d'échange d'information structuré qui repose sur le langage eXtensible Markup Language (XML). Bien que ce dernier puisse être utilisé au-dessus de plusieurs autres protocoles tels que Simple Mail Transfer Protocol (SMTP), Transmission Control Protocol (TCP) ou User Datagram Protocol (UDP), il est majoritairement employé comme couche supérieure à HyperText Transfer Protocol (HTTP). Le protocole SOAP appartient au modèle client/serveur et permet aussi bien l'appel de procédures respectant les propriétés ACID (Atomicité, Cohérence, Isolation et Durabilité), que le transfert d'informations. Dans ce dernier cas, un message est envoyé au serveur qui va alors traiter l'information et répondre au client.

Un message SOAP est un document XML qui contient un en-tête (*header*) ainsi qu'un corps (*body*), le tout encapsulé dans une enveloppe. Cette dernière indique que le document XML correspond à un message SOAP et identifie le début ainsi que la fin de du message.

L'en-tête, qui demeure facultatif, peut contenir différents attributs relatifs au message. Le corps, quant à lui obligatoire, contient les informations soit de la requête faite par le client, soit de la réponse du serveur, ainsi que des informations à propos des erreurs qui pourraient survenir. Les informations contenues dans le corps du message SOAP sont organisées sous forme de blocs. La Figure 3.5 montre un exemple d'échange de messages SOAP à travers HTTP qui permet d'obtenir la valeur du capteur cardiaque d'un dispositif quelconque.

```
POST /Sensors HTTP/1.0
Host: www.liara.uqac.ca
Content-Type: text/xml; charset = utf-8
Content-Length: nnn

<?xml version = "1.0"?>
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV = "http://www.w3.org/2001/12/soap-envelope"
    SOAP-ENV:encodingStyle = "http://www.w3.org/2001/12/soap-encoding">
    <SOAP-ENV:Body xmlns:m = "http://www.liara.uqac.ca/sensors">
        <m:GetSensorValue>
            <m:Sensor>HeartRate</m:Sensor>
        </m:GetSensorValue>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

(a) Requête SOAP

```
HTTP/1.0 200 OK
Content-Type: text/xml; charset = utf-8
Content-Length: nnn

<?xml version = "1.0"?>
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV = "http://www.w3.org/2001/12/soap-envelope"
    SOAP-ENV:encodingStyle = "http://www.w3.org/2001/12/soap-encoding">
    <SOAP-ENV:Body xmlns:m = "http://www.liara.uqac.ca/sensors">
        <m:GetSensorValueResponse>
            <m:HeartRateValue>90</m:HeartRateValue>
        </m:GetSensorValueResponse>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

(b) Réponse SOAP

Figure 3.5 : Exemple d'un échange de messages SOAP entre un client et un serveur pour obtenir la valeur du capteur cardiaque.

L'avantage principal de l'utilisation de SOAP comme méthode de communication est principalement la sécurité qu'offre ce protocole. En effet, ce dernier peut être implémenté par dessus une couche de chiffrement Secure Sockets Layer (SSL), mais il permet également d'exploiter la spécification *Web Services Security ou WS-Security* qui apporte des fonctionnalités de sécurité supplémentaires appréciées des entreprises. Cependant, il demeure un protocole peu flexible et relativement lourd et complexe à mettre en place.

L'ARCHITECTURE REST

Dès le début des années 2000, l'introduction des architectures de type REST (Fielding, 2000) a permis de remplacer petit à petit le protocole SOAP, venant ainsi combler certaines de ses lacunes. Le principal avantage de ces architectures est leur indépendance vis-à-vis des protocoles existants. En effet, bien que les architectures REST soient aussi majoritairement définies par-dessus HTTP, elles pourraient tout autant l'être sur n'importe quel autre protocole—tant que celui-ci admet un schéma d'Uniform Resource Identifier (URI) normalisé. Tout comme pour SOAP, les architectures REST s'appuient sur le modèle de communication client/serveur. Cependant, à l'inverse de SOAP, un service web REST est sans état, c'est-à-dire que le serveur ne connaît pas l'état de chaque client entre les requêtes qu'il doit traiter. Ainsi, du point de vue du serveur, chaque requête est une entité distincte des autres. De plus, contrairement à SOAP, qui impose le format d'échange de données en XML, REST accepte que les ressources, c'est-à-dire, les données échangées entre le client et le serveur, soient exprimées selon plusieurs formats de données tels que XML, JavaScript Object Notation (JSON) ou

encore Hypertext Markup Language (HTML). Ceci garantit alors un meilleur support pour les clients.

Lorsqu'elle est définie sur HTTP, une architecture REST manipule ses ressources avec les différentes méthodes HTTP (`GET`, `POST`, `PUT`, `DELETE`, `PATCH`). Ainsi, lorsqu'un client émet une requête indiquant l'opération qu'il souhaite effectuer sur une ressource donnée, la réponse transmise par le serveur contient alors deux éléments importants—l'en-tête et le corps de la réponse. Plus précisement, l'en-tête contient des informations importantes à propos de l'échange d'information :

- La version du protocole HTTP utilisé pour le transport des données.
- Le code HTTP (Fielding et Reschke, 2014) qui indique l'état de la réponse.
- Le format de donnée.
- *etc.*

Le corps de la réponse, quant à lui, peut contenir une ressource, un message d'erreur, *etc.* La Figure 3.6 illustre le processus de communication entre un client et un serveur dans une architecture REST basée sur les méthodes HTTP. Le client demande l'obtention de la valeur du capteur cardiaque d'un dispositif quelconque. Cette ressource est identifiée par l'URI : `http://liara.uqac.ca/sensors/health/1`. Puisque le serveur est capable de fournir la réponse attendue, et ce, sans erreur, le code HTTP 200, ainsi que le nom et la valeur du capteur sont renvoyés au client au format JSON, tel que précisé dans l'en-tête de la réponse.

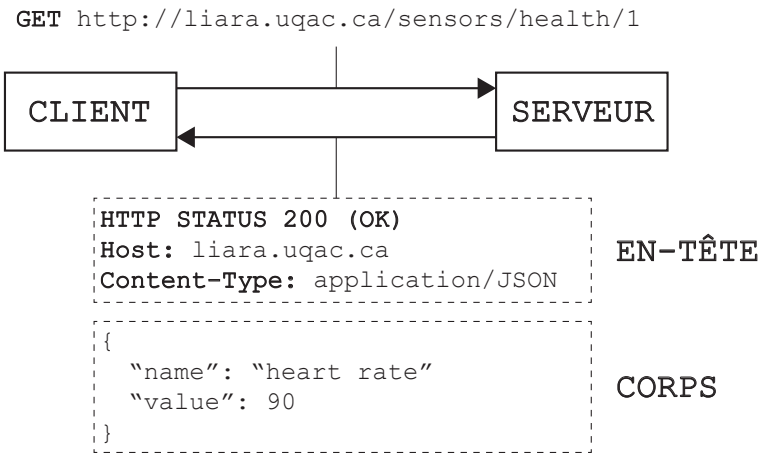


Figure 3.6 : Exemple d'un échange de messages entre un client et un serveur dans une architecture REST basée sur HTTP qui permet de récupérer la valeur d'un capteur cardiaque.

LE PROTOCOLE CoAP

Plus récemment, la recrudescence du nombre d'objets connectés et plus particulièrement de *wearable devices* a fait émerger plusieurs problématiques inhérentes à la limitation de leurs ressources telles qu'une puissance de calcul restreinte, une faible quantité de mémoire ou encore une autonomie réduite. Cependant, la limitation de ces dispositifs implique également de nouvelles problématiques au regard du processus d'échange de données. En effet, malgré une meilleure efficacité qu'avec le protocole SOAP, tant en termes de débit qu'en termes de consommation de ressources, l'exploitation des services web à travers une architecture REST s'est parfois montrée inadaptée dans le contexte de l'IoT (Kovatsch *et al.*, 2011). En effet, lorsqu'une architecture REST repose sur le protocole HTTP, il a été observé qu'une grande fréquence de requêtes ne retournant qu'un nombre faible de données entraînait une surcharge réseau, principalement à cause des en-têtes HTTP qui sont encodés en American

Standard Code for Information Interchange (ASCII) (Shelby, 2010). Néanmoins ce cas de figure correspond au fonctionnement typique d'un réseau de capteurs sans-fil, où chaque message qui est envoyé correspond généralement à une seule mesure. Par conséquent, le protocole CoAP a été introduit par Shelby *et al.* (2014) afin de mieux répondre aux besoins des appareils ayant de fortes contraintes liées au débit, à la consommation et à la puissance de calcul. Malgré son jeune âge, plusieurs recherches se sont tournées vers une implantation de CoAP au sein des habitats intelligents (Bergmann *et al.*, 2012; Mainetti *et al.*, 2015). Par ailleurs, de récents travaux, comme celui introduit par Plantevin *et al.* (2017), ont proposé de nouveaux protocoles de communication qui s'inspirent de la spécification de CoAP. Bien qu'ils soient encore très peu exploités, ceux-ci visent principalement à mieux intégrer l'IoT au sein des habitats intelligents, puisqu'ils permettent d'optimiser davantage le processus d'échange de données.

Pour mettre en place une architecture REST, CoAP reprend plusieurs concepts de HTTP comme les échanges de messages asynchrones, par exemple. Cependant plusieurs optimisations ont été faites pour le rendre plus adapté aux systèmes embarqués tel que la nécessité d'utiliser le protocole UDP qui permet de se passer des mécanismes de fiabilité qui sont obligatoire avec le protocole TCP lors d'échanges de messages. En outre, les en-têtes sont compressés afin de réduire la complexité du décodage et les besoins en bande passante. Les messages CoAP contiennent les informations suivantes :

- La version du protocole utilisée pour le transport des données.

- Le type du message indique s'il s'agit, d'un message fiable qui exige un acquittement (`CON`), d'un message asynchrone qui n'a pas besoin d'être acquitté (`NON`), d'un message d'acquittement, c'est-à-dire, une réponse à une requête `CON` (`ACK`), d'un message qui indique que le serveur a bien reçu la requête, mais qu'il n'a pas le contexte nécessaire pour fournir une réponse (`RST`).
- Le nombre d'options transmises dans l'en-tête (`OC`).
- Un code qui indique si le message est une requête, une réponse ou un message vide. Dans le cas d'une requête, la méthode utilisée est également précisée. Elles sont identiques aux méthodes HTTP.
- Un identifiant unique pour détecter les messages dupliqués et pour faire la correspondance entre un message `CON` et son `ACK` respectif.
- Différentes options, par exemple, un paramètre pour définir la durée de validité des données transmises.
- Les données en elles-mêmes.

La Figure 3.7 illustre le processus de communication entre un client et un serveur dans une architecture REST basée sur CoAP. Le client demande l'obtention de la valeur du capteur cardiaque d'un dispositif quelconque où la requête et la ressource sont respectivement identifiées par le code `0x4d45` et l'URI `/sensors/health/hr`. Le serveur transmet alors immédiatement le message `ACK` ainsi que la valeur du capteur.

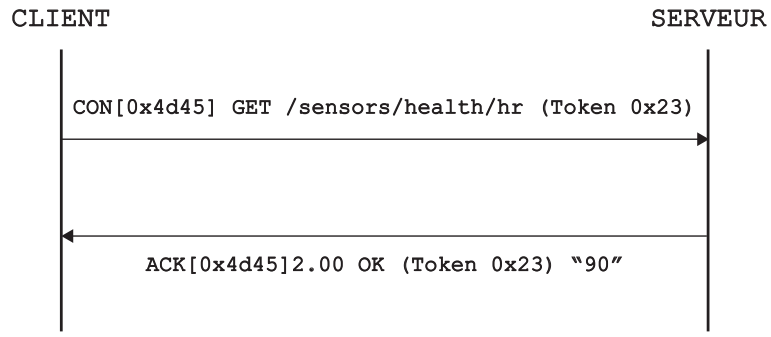


Figure 3.7 : Exemple d'un échange de messages entre un client et un serveur dans une architecture REST basée sur CoAP qui permet de récupérer la valeur d'un capteur cardiaque.

Puisque la différence majeure entre CoAP et HTTP réside dans l'utilisation du protocole UDP, sous-jacent, ceci lui permet de supporter le *multicast*. Aussi, comme CoAP est principalement conçu pour l'IoT, ce protocole se base principalement sur IPv6 bien que, dans de plus rare cas, il puisse également exploiter IPv4. La communication à travers IPv6 permet donc à ce protocole de mieux prendre en compte le volume d'entités qui composent un réseau de capteurs sans-fil. Enfin, il permet de mettre en place aussi bien un modèle client/serveur, tel qu'illustré dans l'exemple précédent, qu'un modèle *publish/subscribe* à l'instar du protocole MQTT.

3.5 CONCLUSION

Dans un premier temps, ce chapitre s'est intéressé à la couche matérielle qui compose les *wearable devices*. Les capteurs les plus adéquats pour être embarqués dans ce type de dispositifs ont été regroupés en différentes catégories qui sont : les capteurs de mouvement, les capteurs physiologiques, les capteurs de courbure et de force ainsi que les capteurs

environnementaux. Bien qu'il en existe beaucoup d'autres, ce chapitre a montré que ceux-ci demeurent les plus utilisés dans divers domaines de recherche tels que la réhabilitation, la surveillance de la santé et surtout dans la reconnaissance de gestes et d'activités.

Ensuite, ce chapitre a présenté les technologies de communication sans-fil actuellement employées par les *wearable devices*, mais également les futures évolutions de chacune d'elles. Il est apparu que toutes s'inscrivent dans l'optique de mieux prendre en compte le nombre grandissant d'objets connectés, plus particulièrement en ce qui concerne leur limitation énergétique. De plus, en ne considérant que les capacités actuelles de chaque technologie, ce chapitre a proposé un comparatif de leurs différences en termes de capacité de portée, débit et robustesse. Ceci a permis de conclure que le choix de la technologie à adopter dans le processus de conception de *wearable devices* doit se faire en fonction des contraintes liées à leur utilisation.

Enfin, dans sa dernière partie, ce troisième chapitre a exposé les différents protocoles et architecture de haut niveau permettant aux *wearable devices* d'échanger leurs données avec d'autres entités qui composent un réseau (capteurs intelligents, serveur central, *etc.*). Ainsi, dans les travaux sur les habitats intelligents et leurs évolutions pour y intégrer l'IoT, deux principaux modèles pour l'échange de données ont été retenus. Le premier est le modèle *publish/subscribe* et plus particulièrement le protocole MQTT. Le second concerne le modèle client/serveur, plus traditionnel. Bien qu'il admette un fonctionnement particulier, le cas du BLE s'appuie fortement sur ce deuxième modèle. De plus, des méthodes plus récentes issues du web social, comme la consommation de services web, appartiennent également à

un modèle client/serveur. Pour tirer profit des services web, plusieurs technologies comme le protocole SOAP ou les architectures REST ont été mises en place au sein des infrastructures d'habitats intelligents. Cependant, certaines limitations quant à l'intégration de l'IoT ont pu être observées. Pour combler ces problématiques, de nouvelles implémentations comme le protocole CoAP ont été mises en place dans ces habitats.

CHAPITRE IV

UN *WEARABLE DEVICE* POUR LA RECONNAISSANCE DES SOLS

Dans le chapitre précédent, les différentes technologies aussi bien matérielles que logicielles qui sont utilisées dans la conception des *wearable devices* ont été présentées. Puisque ces technologies sont nombreuses, il demeure donc important de bien cibler celles qui doivent être utilisées afin que la conception des *wearable devices* réponde à la problématique exprimée. Ainsi, la principale question qui est traitée dans ce chapitre est : *« quels sont les nouveaux apports, en matière d'Intelligence Ambiante, que les wearable devices vont permettre de proposer aux résidents des habitats intelligents afin d'améliorer l'assistance qui leur est requise ? »*.

Pour répondre à cette question, notre travail s'est concentré sur un exemple précis d'utilisation des *wearable devices* : la reconnaissance des différents types de sols à l'aide de données inertielles produites par la démarche humaine. Ce cas d'application est directement lié à la problématique générale de l'assistance, car au sein des habitats intelligents, les résidents doivent constamment s'adapter à différents types de sols. Dans ce contexte, lorsqu'il s'agit de personnes en perte d'autonomie ou ayant des troubles moteurs, certains sols peuvent alors représenter des dangers ou causer de la peur chez les résidents, car il est possible que ces sols soient plus ou moins meubles ou glissants. Par exemple, il est possible de mentionner les tapis ou le carrelage mouillé dans la salle de bain.

De manière générale, très peu de travaux proposant une méthode de reconnaissance des types de sols ont été identifiées dans la littérature. Ainsi, la première partie de ce chapitre présente brièvement les méthodes qui s'appuient sur des données inertielles produites par des robots. Ensuite, après avoir présenté la solution proposée ainsi que les expérimentations réalisées, ce chapitre va analyser et discuter les résultats obtenus. Finalement, la dernière partie dresse une conclusion de ce premier travail.

4.1 LA RECONNAISSANCE DES TYPES DE SOLS

Dans le domaine de la reconnaissance des types de sols, la majorité des travaux que nous avons recensés concerne des méthodes qui ont été développées pour être utilisées avec des robots. Bien que ceux-ci soient, d'une certaine manière, pertinents pour comprendre comment reconnaître différents types de sols, ces méthodes ne sont pas directement adaptées pour effectuer, de façon appropriée, une telle reconnaissance dans le contexte de la démarche humaine. Par ailleurs, la compréhension de ces travaux a permis de guider les choix de conceptions de la méthode et du protocole expérimental qui sont proposés dans ce chapitre.

Dans le domaine de la robotique, l'idée de reconnaître des types de sols, et plus particulièrement, par le biais de données produites par une centrale inertielle n'est pas nouvelle. Tout d'abord, Vail et Veloso (2004) ont expérimenté la détection de surface grâce à des données inertielles produites par un robot quadrupède. Pour réaliser l'apprentissage, les auteurs ont opté pour un algorithme dont le modèle est un arbre de décision (`C4.5`), car ils estiment qu'il s'agit d'un algorithme suffisamment rapide et facile à représenter et à implémenter. Pour

quantifier la précision de leur modèle d'apprentissage, Vail et Veloso ont utilisé la technique de la validation croisée en 10-plis (Kohavi, 1995), ce qui leur a permis d'obtenir un taux de reconnaissance global de 84.9% (ciment : 91% ; tapis : 81.2% ; champs : 81.2%). Par la suite, Kertesz (2016) a également présenté une méthode permettant de reconnaître des sols avec un accéléromètre embarqué sur un robot quadrupède. Un taux de reconnaissance de 96.2% a été obtenu grâce à une forêt d'arbres décisionnels (Random Forest ou RF), où la précision de l'apprentissage a été évaluée avec la même technique que celle utilisé par Vail et Veloso. La différence majeure entre les deux travaux réside principalement dans le fait que les premiers ont exploités des caractéristiques temporelles du signal inertiel (variance et correlation) pour concevoir leur modèle d'apprentissage, tandis que le second a employé des caractéristiques fréquentielles obtenues grâce à une FFT.

Par ailleurs, Bibuli *et al.* (2007) ont proposé une méthode de reconnaissance des sols pour un robot à quatre roues équipé de plusieurs types de capteurs, dont une centrale inertielle. Pour ce faire, ils ont tout d'abord calculé les composantes fréquentielles du signal inertiel *via* l'algorithme de la transformée de Fourrier discrète (Discrete Fourier Transform ou DFT), pour chacun des axes du capteur. Ensuite, chaque ensemble de données résultant de cette transformation a été entraîné distinctement par un réseau de neurones artificiels (ANN). Par conséquent, les meilleurs résultats ont été obtenus avec les données de l'axe x du gyroscope, soit respectivement 90%, 71.2%, 70%, 98.8% et 83.5% pour les sols en graviers, gazon, sable, pavés et terre.

Enfin, Weiss *et al.* (2007) ont proposé une comparaison de plusieurs méthodes pour réaliser une reconnaissance de sols avec un robot à quatre roues doté d'un capteur inertiel. Plus précisément, ils ont comparé l'algorithme SVM avec d'autres types d'algorithmes d'apprentissage où les données fournies en entrée sont des caractéristiques fréquentielles du signal inertiel correspondant à différentes vitesses du robot (0.2, 0.4 et 0.6 m/s) et pour six sols distincts (sol intérieur, asphalte, gravier, gazon, pavés, sol argileux). Avec l'obtention de 77% de taux de confiance, leur étude a montré de meilleurs résultats avec l'algorithme SVM qu'avec les autres algorithmes qui sont : un réseau de neurones probabiliste (Probabilistic Neural Network ou PNN), l'algorithme des k plus proches voisins, l'algorithme bayésien naïf et `C4.5`.

Au mieux de notre connaissance, il apparait que peu de recherches ont été proposées pour exploiter une telle reconnaissance avec l'humain. Pourtant, de nombreux cas d'utilisation à la reconnaissance des sols dans ce contexte nous paraissent exploitables. En effet, dans l'objectif de proposer une meilleure assistance aux résidents des habitats intelligents, il semble important d'apporter des réponses à cette problématique spécifique. Par exemple, des techniques de prévention des chutes plus efficaces permettraient alors de rassurer et d'aider les personnes en perte d'autonomie ou ayant des troubles moteurs lors de leur évolution au sein de ce genre d'habitats.

De ce fait, Otis *et al.* (2016), au travers de leurs travaux sur la stimulation vibrotactile pour la réduction des risques de chutes, ont récemment proposé une méthode basée sur la reconnaissance des types de sols. Pour ce faire, une chaussure embarquant un accéléromètre a été fabriquée. Ensuite, un algorithme de *clustering* a été utilisé pour segmenter les caractéristiques

fréquentielles discriminantes obtenues par le biais de la FFT sur le signal accéléromètrique lorsque celui-ci a été enregistré sur les sols suivants : gravier, ciment sable, neige et glace. En ce qui concerne les résultats obtenus, les auteurs mentionnent un taux d'erreurs compris entre 1% et 5% lors des essais en laboratoire, alors qu'ils ont observé une augmentation de ce taux en conditions réelles, soit 20% d'erreurs. Malgré l'absence de détails supplémentaires sur l'obtention de ces résultats, le travail réalisé par Otis *et al.* (2016) nous a permis d'identifier un protocole expérimental adapté à mettre en œuvre. Ceci dans le but d'être le plus fidèle aux conditions réeles d'utilisation, ainsi que pour faciliter la comparaison de la méthode proposée dans ce chapitre avec de futures implémentations.

4.2 SOLUTION PROPOSÉE

4.2.1 LE *WEARABLE DEVICE*

Selon la littérature existante, l'utilisation de centrales inertielles pour réaliser la reconnaissance de sols s'est révélée être une technique aussi bien fonctionnelle avec des robots qu'avec les individus. De ce fait, la contribution présentée dans ce chapitre s'est, dans un premier temps, axée sur la conception d'un *wearable device* pour y parvenir. L'idée générale est de piloter, par l'intermédiaire d'un téléphone connecté en BLE (`cell_oper`), le dispositif afin d'enregistrer un volume important de données pendant une marche. Lors de la conception, le choix s'est porté sur le système sur puce (System on Chip ou SoC) *Arduino 101* qui embarque le module *Intel Curie*, également accompagné d'un *shield* de prototypage sur lequel est embarqué un lecteur de carte mémoire, tel qu'illustré en Figure 4.1. Ce choix a été motivé par

la composition même du module *Intel Curie*. En effet, ce dernier intègre d'usine un module de communication BLE, un IMU 6-axes (6-Degrees of Freedom ou 6-DoF) ainsi qu'une quantité suffisante d'Entrées/Sorties (Input/Output ou I/O) pour connecter des composants supplémentaires et ainsi faciliter l'évolution du dispositif. De plus, les autres avantages au choix de ce matériel demeurent la simplicité et la fiabilité de l'*Arduino 101*, mais également la quantité de ressources disponibles pour cette plateforme. Finalement, la présence d'un lecteur de carte mémoire sur le *shield* de prototypage constitue un moyen simple et efficace pour gérer la quantité de données inertielles qui doivent être stockées.

Dans une première version, la reconnaissance des types de sols a été réalisée par l'intermédiaire de l'IMU 6-axes directement embarqué sur le SoC *Arduino*. Néanmoins, puisque la précision de celui-ci était inconnue avant les premières expérimentations et qu'il manquait les trois axes supplémentaires du magnétomètre, une seconde version du *wearable*

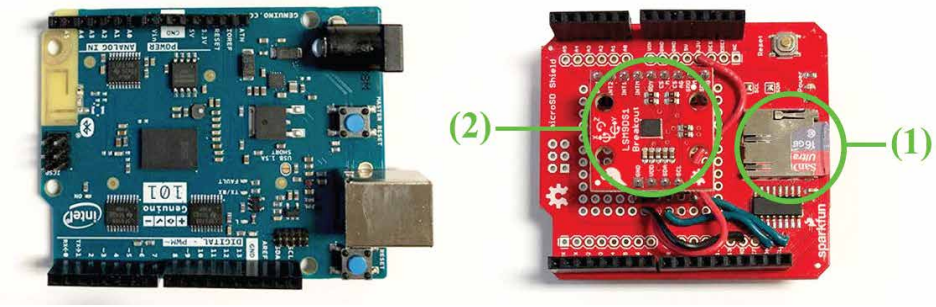


Figure 4.1 : SoC Arduino 101 et son *shield* de prototypage incluant une carte mémoire (1) ainsi que la centrale inertielle *LSM9DS1* (2) présente sur la version 2 du dispositif (Thullier *et al.*, 2017).

device a été proposée. Pour cette dernière, un nouvel IMU 9-axes, le `LSM9DS1`, a été interfacé directement sur le *shield* de prototypage.

4.2.2 LE *FIRMWARE*

Pour assurer le bon fonctionnement du matériel qui compose le *wearable device*, il était tout d'abord nécessaire de développer un *firmware* [14]. Le fonctionnement de celui-ci s'articule autour de trois composants logiciels principaux qui sont illustrés en Figure 4.2.

Le premier est le module de `commande`. Il est en charge des communications entre le téléphone (`cell_oper`) et le *wearable device*, par l'intermédiaire du module radio BLE. Ainsi, un téléphone ou n'importe quel autre dispositif compatible avec la technologie BLE peut être utilisé pour étiqueter les ensembles de données enregistrés avec les informations de configuration requises (le type de sol, l'identifiant anonyme du participant et l'emplacement du dispositif). Le second module principal est l'`IMU Recorder`. En fonction de la version du dispositif, il a pour rôle de stocker dans une mémoire tampon les valeurs de l'IMU qui transitent soit par un bus de données Serial Peripheral Interface (SPI) dans sa première version, soit *via* un bus de données Inter-Integrated Circuit (I^2C), à une fréquence stabilisée à 60 Hz. Le contenu de la mémoire tampon est ensuite envoyé au module de `formatage des données` toutes les secondes. Finalement, ce composant récupère l'ensemble des données de configuration du module de commande et s'occupe d'écrire les nouvelles données dans un fichier Comma-Separated Values (CSV) qui est stocké sur la carte mémoire. Dans la seconde

14. https://github.com/FlorentinTh/SoilTypesRecognition-WearableFirmware

version du *wearable device*, le module de `formatage des données` s'occupe également de calculer les angles d'Euler (précession, nutation et rotation propre) grâce aux trois axes supplémentaires fournis par le `LSM9DS1`.

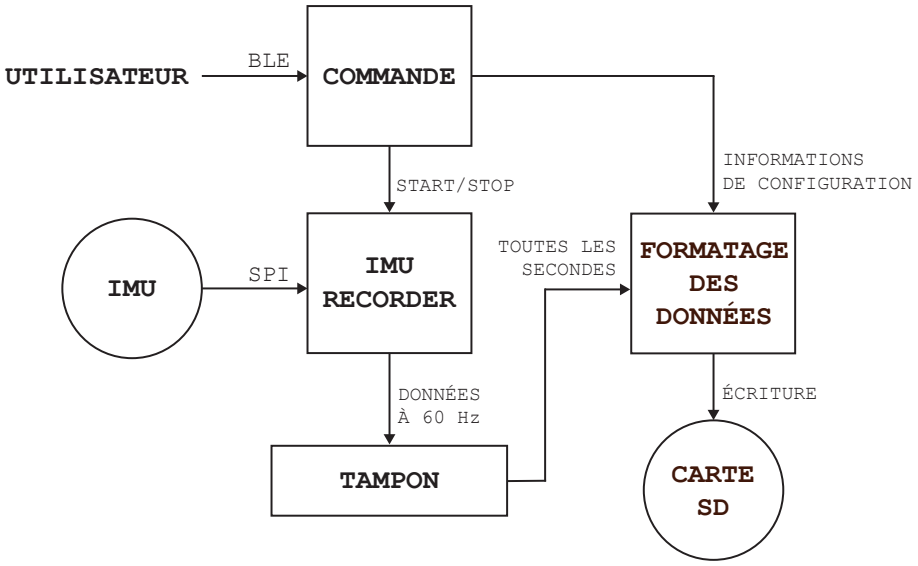


Figure 4.2 : Implémentation du *firmware* embarqué sur le *wearable device* (Thullier *et al.*, 2017).

4.2.3 LE PROCESSUS D'APPRENTISSAGE POUR LA RECONNAISSANCE DES TYPES DE SOLS

EXTRACTION DES CARACTÉRISTIQUES

Après que les données aient été enregistrées sur la carte mémoire, le processus d'apprentissage pour la reconnaissance des types de sol peut débuter et sa première phase demeure l'extraction des caractéristiques. En ce sens, cette solution propose d'utiliser les caractéris-

tiques temporelles et fréquentielles les plus employées dans le domaine de la reconnaissance d'activités, tel que discuté à la Section 2.2.1 et qui sont rappelées par le Tableau 4.1.

La première opération consiste en une simple moyenne non pondérée sur chacun des axes tant « physiques » des capteurs de l'IMU (gyroscope, accéléromètre et magnétomètre si 9-axes) que « logiques » (angles d'Euler), s'ils existent. Ensuite, la moyenne générale de chacune de ces caractéristiques est calculée pour l'ensemble des axes. De la même manière, cette procédure a été appliquée à plusieurs autres calculs statistiques : l'écart type, l'asymétrie (Équation 2.6), l'aplatissement (Équation 2.7), le *Zero Crossing Rate* et la corrélation entre toutes les combinaisons possibles d'axes par capteurs de l'IMU (Équation 2.8). Dans un second temps, une transformation du domaine temporel vers le domaine fréquentiel a été nécessaire pour obtenir les caractéristiques suivantes : la composante continue, l'énergie spectrale (Équation 2.9) et l'entropie (Équation 2.10). Ce passage d'un domaine à l'autre a été réalisée en utilisant la transformation de Fourier rapide (FFT) selon l'algorithme de Bluestein (Bluestein, 1970). Ainsi, en fonction de la centrale inertielle exploitée (6-axes, 9-axes ou 9-axes avec les trois angles d'Euler), ce sont 70, 105 ou 140 caractéristiques qui sont calculées sur chaque fenêtre temporelle rectangulaire et non chevauchante, dont la taille est fixée à 60 secondes. La taille de la fenêtre a été déterminée par une évaluation du temps moyen nécessaire pour marcher une distance de 4.3 m, ce qui correspond à un aller-retour dans le bac utilisé lors des expérimentations et dont le protocole est décrit à la Section 4.3.

Tableau 4.1 : Récapitulatif de toutes les caractéristiques utilisées par la solution proposée en fonction du nombre d'axes offerts par la centrale inertielle.

Domaine Temporel				**Domaine Fréquentiel**			
Caractéristique	***Nb. Total de Caractéristiques***			***Caractéristique***	***Nb. Total de Caractéristiques***		
	IMU 6 axes	*IMU 9 axes*	*IMU 9 axes avec angles d'Euler*		*IMU 6 axes*	*IMU 9 axes*	*IMU 9 axes avec angles d'Euler*
Moyenne pour chaque axe	6	9	12	Composante continue pour chaque axe	6	9	12
Moyenne de tous les axes	2	3	4	Énergie spectrale pour chaque axe (Équation 2.9)	6	9	12
Écart type pour chaque axe	6	9	12	Entropie pour chaque axe (Équation 2.10)	6	9	12
Écart type pour tous les axes	2	3	4	-	-	-	-
Asymétrie pour chaque axe (Équation 2.6)	6	9	12	-	-	-	-
Asymétrie pour tous les axes	2	3	4	-	-	-	-
Aplatissement pour chaque axe (Équation 2.7)	6	9	12	-	-	-	-
Aplatissement pour tous les axes	2	3	4	-	-	-	-
Corrélation entre toutes les combinaisons possibles d'axes incluant le total des axes	12	18	24	-	-	-	-
Zero Crossing Rate pour chaque axe	6	9	12	-	-	-	-
Zero Crossing Rate pour tous les axes	2	3	4	-	-	-	-
Sous-total	***52***	***78***	***104***	***Sous-total***	***18***	***27***	***36***
Total	***IMU 6 axes***	**70**					
	IMU 9 axes	**105**					
	IMU 9 axes avec angles d'Euler	**140**					

APPRENTISSAGE

Dès l'obtention des caractéristiques discriminantes, l'étape suivante requise pour réaliser la reconnaissance des sols est le processus d'apprentissage. Selon la littérature qui concerne la reconnaissance d'activités, mais également celle au sujet de la reconnaissance des sols en

robotique qui ont été respectivement discutées dans les Sections 2.2.2 et 4.1, la performance de plusieurs algorithmes d'apprentissage a pu être démontrée. Néanmoins, ce premier travail propose une comparaison entre deux algorithmes qui n'ont pas encore été présentés dans cette thèse : la forêt d'arbres décisionnels (*random forest*) et les k plus proches voisins (k-Nearest Neighbors ou k-NN). Ce choix est motivé par deux raisons principales. La première est que ces deux techniques d'apprentissage appartiennent à deux familles distinctes—respectivement, les arbres de décision et les algorithmes d'apprentissage « paresseux ». De plus, ceux-ci constituent tous deux des méthodes non paramétriques, c'est-à-dire, qui ne font aucune supposition quant à la distribution de l'échantillon de données fournies en entrée. De ce fait, ces méthodes demeurent simples, flexibles et efficaces (Russell et Norvig, 2010). Par ailleurs, la performance en termes de taux de reconnaissance obtenue avec ces algorithmes est souvent exposée dans la littérature (Kertesz, 2016; Vail et Veloso, 2004).

Selon Breiman (2001), une forêt d'arbres décisionnels est une combinaison d'arbres prédicteurs, aussi appelés arbre de décision, où chacun d'eux dépend d'un vecteur de données distinct, mais de même cardinalité. Ce vecteur contient un sous-ensemble aléatoire des données initiales issues de la phase d'extraction des caractéristiques. La décision finale du processus de reconnaissance de cet algorithme est déterminée par un vote majoritaire entre chacune des décisions émises en sortie des arbres de décision qui composent la forêt. La Figure 4.3 illustre un exemple simplifié d'une forêt d'arbres décisionnels utilisant trois arbres.

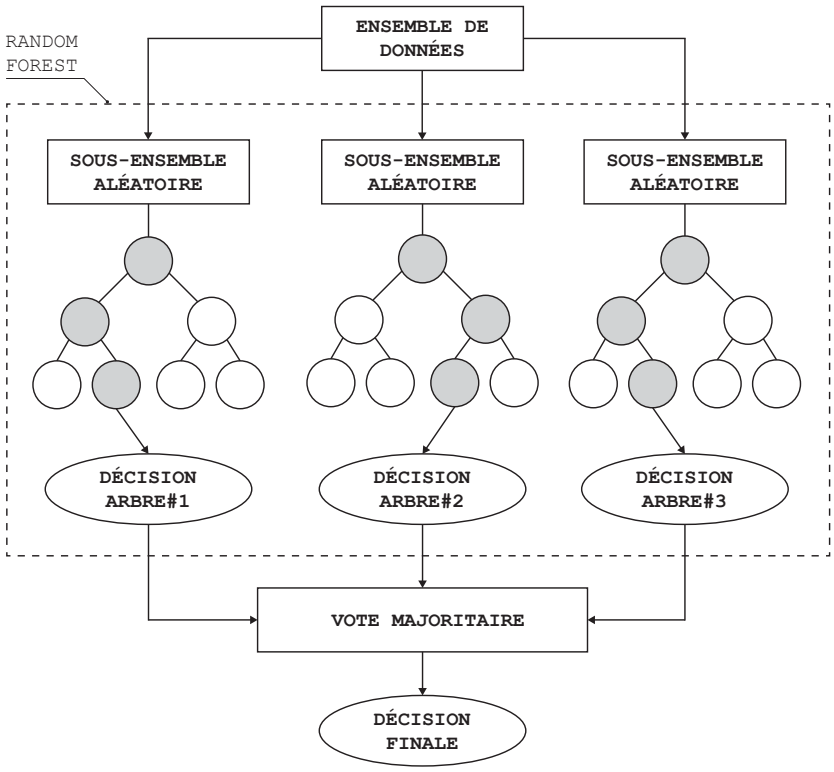


Figure 4.3 : Exemple de l'algorithme *random forest* utilisant $B = 3$ arbres (Thullier *et al.*, 2017).

Pour exploiter la forêt d'arbres décisionnels en tant qu'algorithme d'apprentissage, il est important de définir préalablement au moins trois paramètres principaux. Le premier est le nombre d'arbres que doit comporter la forêt (B). Le second correspond au nombre de caractéristiques à considérer dans la division des nœuds de l'arbre lors de la construction de ce dernier (F) et enfin, la fonction nécessaire à la mesure de la qualité de cette division (C). En ce qui concerne le nombre de caractéristiques, Breiman suggère d'utiliser $F_1 = \lfloor log_2(m) + 1 \rfloor$ où m fait référence au nombre total d'attributs présents dans le jeu de données fourni en entrée. Néanmoins, il est également possible de trouver d'autres recommandations comme $F_0 = \lfloor \frac{1}{2}\sqrt{m} \rfloor$ ou encore $F_2 = \lfloor \sqrt{m} \rfloor$. Par ailleurs, en ce qui concerne la mesure de la qualité de la division des nœuds, les fonctions qui sont les plus couramment utilisées sont le calcul du

cœfficient de Gini et l'évaluation du gain en information basé sur l'entropie de Shannon (E) dont les équations sont respectivement rappelées ci-après :

$$Gini(T) = 1 - \sum_{i=1}^{n} (p_i)^2 \tag{4.1}$$

$$E(T) = \sum_{i=1}^{n} -(p_i log_2 p_i) \tag{4.2}$$

où T correspond aux données qui contiennent les instances de n étiquettes et p_i est la fréquence relative de l'étiquette $i \in n$ dans T. Le choix quant à ce paramètre dépend essentiellement du type d'arbre de décision qui est utilisé lors de la construction de la forêt (*p. ex.* ID3, `C4.5`, CART, *etc.*).

D'autre part, l'algorithme des k plus proches voisins est considéré comme une technique d'apprentissage dite « paresseuse », ce qui veut dire qu'il n'admet pas de phase d'entraînement, ou que celle-ci demeure minime. Bien que cet algorithme soit relativement rapide et flexible, la décision finale d'étiquetage est réalisée en se basant sur l'intégralité du jeu de données utilisé pour l'entraînement. En conséquence, celui-ci doit être stocké en mémoire, ce qui implique alors la nécessité de disposer d'une quantité de stockage importante. En effet, cette méthode suppose que les données peuvent être représentées dans un espace vectoriel de caractéristiques qui peut être multidimensionnel. Ainsi, la phase d'apprentissage de l'algorithme consiste à stocker les vecteurs de données ainsi que les étiquettes qui leur sont associés. Ensuite, lors de

la phase de reconnaissance, il s'agit de déterminer quels sont les k vecteurs de caractéristiques qui sont les plus proches pour chaque nouvelle donnée à étiqueter, où k est un paramètre qui doit être défini préalablement. Ces plus proches voisins sont alors extraits grâce à une fonction de mesure qui doit également être déterminée au préalable (*p. ex.* la distance euclidienne : D_e ou la distance de Manhattan : D_m respectivement données en Équations 2.13 et 2.14). Finalement, l'étiquette de la nouvelle donnée est alors attribuée selon un vote majoritaire entre celles de chaque k plus proche voisin. La Figure 4.4 illustre un exemple de cette méthode d'apprentissage avec $k = 3$ plus proches voisins.

4.3 EXPÉRIMENTATIONS

Les expérimentations mises en place pour valider le système de reconnaissance des sols se sont déroulées en deux étapes et à deux périodes de l'année différentes. La première a été réalisée en hiver, avec la première version du *wearable device* : `wear_v1`. Elle a impliqué neuf étudiants universitaires, tous des hommes ayant entre 22 et 36 ans. Leurs poids se situaient entre 65 kg et 110 kg (poids médian : 80 kg) et leurs tailles étaient comprises entre 172 cm et 192 cm (taille médiane : 183 cm). Parmi ceux-ci, il y avait 7 droitiers pour 2 gauchers et tous étaient en bonne santé sans aucun problème de motricité. La seconde étape, quant à elle, s'est

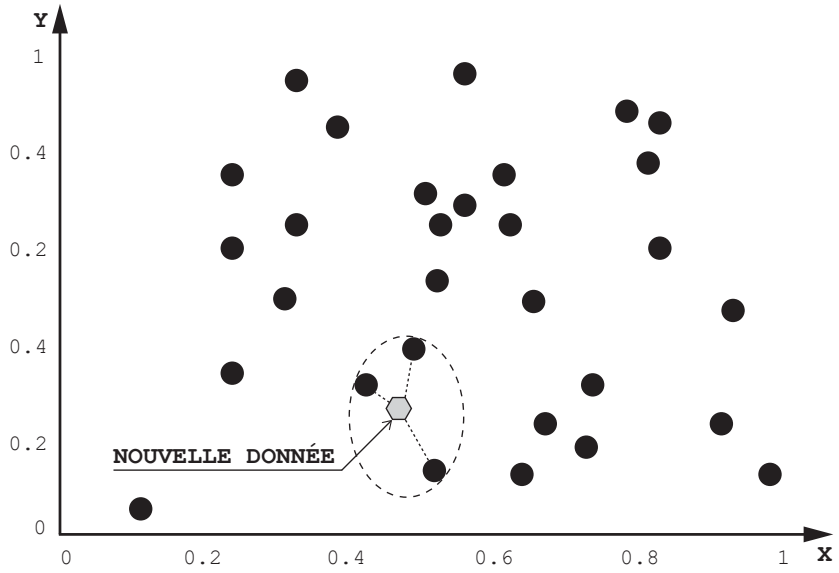


Figure 4.4 : Exemple de l'algorithme des k plus proches voisins où $k = 3$ (Thullier *et al.*, 2017).

déroulée en été avec la deuxième version du dispositif : `wear_v2` et un téléphone intelligent : `cell` (*Huawei Nexus 6P* avec la version 8.0 d'*Android*). Cependant, seulement six participants sur les neuf de la première étape ont été en mesure de mener à bien cette seconde expérimentation en raison de la fin de leur cursus universitaire.

4.3.1 MISE EN ŒUVRE

Dans un premier temps, puisque l'UQAC se situe dans une région où les conditions météorologiques peuvent être difficiles, la plupart des différents types de sols sont, en fonction de la période de l'année, recouverts de glace ou de neige. Pour remédier à cette problématique, un bac a été fabriqué afin d'assurer le déroulement des expérimentations à l'intérieur du laboratoire. Par conséquent, le contenu de celui-ci était soit du gravier, soit du sable, tel qu'illustré par la Figure 4.5.

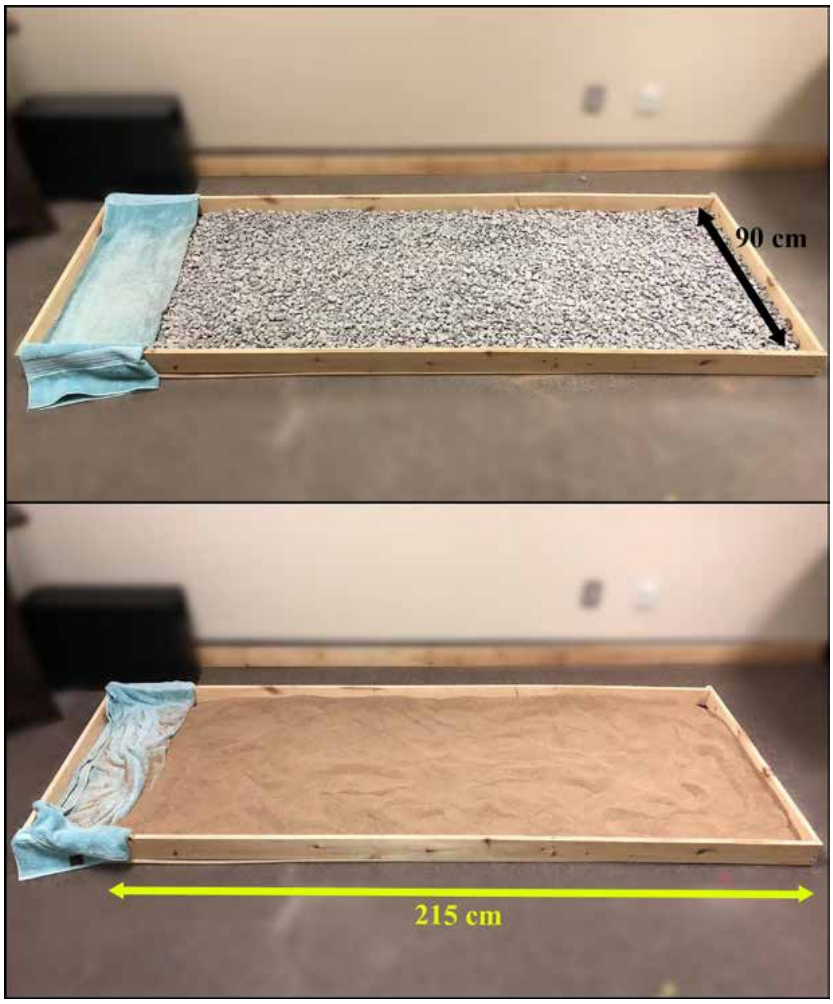


Figure 4.5 : Bac utilisé lors des expérimentations rempli de haut en bas, de gravier et de sable (dimensions : $l : 215\ cm \times L : 90\ cm \times H : 15\ cm$**) (Thullier *et al.*, 2017).**

Dans un deuxième temps, une application [15] *Android* a été développée afin de piloter les expérimentations et par conséquent, étiqueter les jeux de données pour chacun des enregistrements produits par les participants. Celle-ci a permis de définir les temps de début et de fin, ainsi que les différentes informations de configuration (le type de sol, l'identifiant anonyme du participant et l'emplacement du dispositif) par le biais d'un téléphone intelligent

15. https://github.com/FlorentinTh/SoilTypesRecognition-AndroidAppDriver

spécifiquement dédié à cette tâche : `cell_oper` (*LG Nexus 5* avec la version 6.0.1 d'*Android*). Puisque le *wearable device* est doté d'une connectivité BLE, la première opération a consisté en un balayage afin d'obtenir la liste des appareils émetteurs aux alentours. Dans notre cas, `cell_oper` est le dispositif périphérique et le *wearable device* est le dispositif central. Ensuite, dès que les deux appareils ont été couplés, le *wearable device* est alors devenu le serveur GATT et `cell_oper` le client, tel qu'illustré par le premier ecran de la Figure 4.6. Enfin, l'enregistrement des données sur le *wearable device* a débuté dès lors que les informations de configuration ont été renseignées et que le bouton d'envoi des données a été appuyé, comme le montre le second écran de la Figure 4.6.

L'avantage principal de cette méthode réside principalement dans la mobilité qu'offre l'utilisation d'un téléphone intelligent par rapport à un ordinateur traditionnel—ce qui a fortement facilité la tâche de supervision des expérimentations. De plus, l'application a pu être réutilisée sans subir de modifications aussi bien avec les deux versions du *wearable device* (`wear_v1` et `wear_v2`) qu'avec le téléphone intelligent utilisé pour l'enregistrement de données (`cell`).

Par ailleurs, le fonctionnement de la seconde application [16] *Android* qui a été développée demeure identique à celui du firmware du *wearable device*. En effet, celle-ci reçoit par BLE les mêmes informations de configuration envoyées par le biais de l'application qui s'exécute sur `cell_oper`. De la même manière, les données inertielles (accéléromètre, gyroscope,

16. https://github.com/FlorentinTh/SoilTypesRecognition-AndroidAppRecording

magnétomètre) ainsi que les angles d'Euler sont enregistrés toutes les secondes sur la mémoire flash du téléphone sur lequel elle est exécutée (`cell`), au format CSV et à une même fréquence stabilisée à 60 Hz.

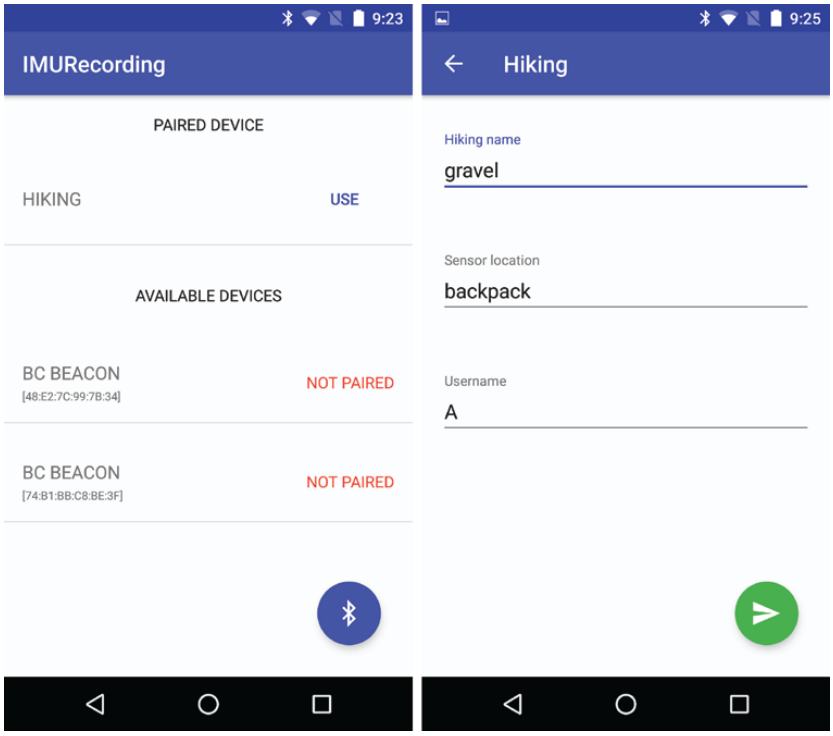


Figure 4.6 : Captures d'écrans de l'application *Android* exécutée sur `cell_oper` qui a permis de le pilotage et l'étiquetage les données brutes (Thullier *et al.*, 2017).

4.3.2 PROCÉDURE

Pour effectuer la collecte de données, il était nécessaire que les expérimentations soient conduites dans des situations se rapprochant le plus possible de cas d'utilisation réels. En ce sens, trois emplacements ont été adoptés comme étant les plus probables d'être choisis par les utilisateurs du *wearable device*. Le premier est à l'intérieur d'un sac porté sur les épaules grâce

aux anses, comme un sac à main. La seconde possibilité identifiée pour le port du dispositif est à l'intérieur d'un sac à dos et enfin, le dernier positionnement est simplement à l'intérieur de la poche d'une veste. Cependant, puisque ce travail s'intéresse à déterminer avec précision l'impact de l'emplacement du *wearable device* sur la reconnaissance des types de sols, il a été demandé aux participants de l'expérimentation de porter le dispositif à la fois sur l'épaule de droite et l'épaule de gauche pour le sac à main, tel qu'illustré par la Figure 4.7a. Aussi, celui-ci était lesté avec une masse supplémentaire d'un kilogramme. De la même manière, les participants ont été invités à porter le dispositif à l'intérieur des poches à droite et à gauche de la veste, comme montré par la Figure 4.7b, où une attention particulière a été portée sur le fait

(a) Positionnement du *wearable device* à l'intérieur d'un sac bandoulière porté sur les épaules de droite et de gauche.

(b) De gauche à droite, positionnement du *wearable device* dans les poches de droite (1) et de gauche (2) ainsi qu'à l'intérieur d'un sac à dos ordinaire de 20 L.

Figure 4.7 : Les cinq emplacements où le *wearable device* a été positionné pendant l'expérimentation (Thullier *et al.*, 2017).

de conserver la veste fermée—ceci dans l'objectif de réduire le bruit potentiel induit sur les données inertielles par l'oscillation des bras lors de la marche. Finalement, en ce qui concerne le sac à dos, sa contenance était de $20\,L$ et il était lesté avec un poids supplémentaire de $3\,kg$. De plus, le *wearable device* a été positionné dans la même poche du sac et dans la même direction pour tous les participants de l'expérimentation.

Dès lors que les instructions au sujet du positionnement du dispositif ont été présentées aux participants, ils ont été invités à effectuer six allers-retours sur chaque type de sol, pour chaque position donnée. Un superviseur en charge de la gestion du *wearable device* par l'intermédiaire de l'application *Android* dédiée leur a indiqué, à la fois quand commencer et quand s'arrêter de marcher. Tout d'abord, il leur a été demandé de commencer par les graviers placés à l'intérieur du bac décrit précédemment. Dans un second temps, ils ont été invités à effectuer la même opération sur le sol à côté du bac, considéré comme étant une sorte de ciment. Les participants ont ensuite été sollicités à sortir pour effectuer six autres allers et retours dans la neige. Afin de rester cohérents avec la procédure de l'expérimentation réalisée au sein du laboratoire, un ruban à mesurer respectant la longueur du bac a été placé sur la neige pour indiquer aux participants où faire demie tour. Enfin, puisque les graviers à l'intérieur du bac devaient être remplacés, les participants ont été recontactés pour effectuer ultérieurement l'expérimentation sur le sable. Cette même procédure a été répétée aussi bien avec la version 2 du *wearable device* (`wear_v2`), ainsi qu'avec le téléphone intelligent (`cell`) à l'exception faite des enregistrements récoltés sur la neige, puisque la période de l'année durant laquelle cette deuxième partie de l'expérimentation a été conduite ne le permettait pas.

4.4 RÉSULTATS ET DISCUSSION

4.4.1 ENSEMBLES DE DONNÉES

Afin de valider que la méthode de reconnaissance des types de sols proposée dans ce premier travail demeure suffisamment précise et fiable, il a été nécessaire de produire plusieurs jeux de données composés des caractéristiques discriminantes et des informations de configuration qui constituent l'étiquetage des données. L'ensemble des différents jeux de données utilisés dans ces expérimentations sont disponibles en *open source* [17] et la liste de ceux-ci est fournie par le Tableau 4.2.

Tableau 4.2 : Liste détaillée des ensembles de données produits, où les noms sont exprimés avec la notation BNF.

Jeu de données	Description
`soil_type_[ wear_v1 | wear_v2 | cell ]_[ 6 | 9 | 12 ]`	Chaque instance est étiquetée avec le type de sol correspondant (*p. ex.* `sable`).
`soil_type_position_[ wear_v1 | wear_v2 | cell ]_[ 6 | 9 | 12 ]`	Chaque instance est étiquetée avec le type de sol correspondant et la position du wearable device (*p. ex.* `poche_droite_sable`).

Puisque les expérimentations faites par les participants ont permis l'échantillonnage de données brutes à une fréquence stabilisée à 60 Hz, il a tout d'abord été nécessaire de définir la taille de la fenêtre pour réaliser le calcul des caractéristiques discriminantes. Celle-ci a été déterminée de manière empirique selon le temps moyen pour marcher la distance d'un

17. https://github.com/LIARALab/Datasets/tree/master/SoilTypesRecognition

aller-retour dans le bac qui a été observé pour chaque participant. Ainsi, puisque la moyenne de ces temps a été de 6 secondes, c'est ce découpage qui a été utilisé pour procéder au calcul des caractéristiques discriminantes, soit une fenêtre de taille $6\,s \times 60\,Hz = 360$ instances de données brutes pour chaque participant, sur chaque type de sol et pour chaque position. Finalement, ces jeux de données ont été utilisés en entrée des algorithmes d'apprentissage pour la reconnaissance des types de sols.

D'autre part, pour vérifier l'hypothèse d'indépendance de positionnement, c'est-à-dire, la capacité pour le système à reconnaitre les types de sols qu'importe l'emplacement du *wearable device*, des ensembles de données distincts ont été produits pour chaque positionnement et dont les données sont étiquetées avec le type de sol uniquement.

4.4.2 ÉVALUATION DE LA PERFORMANCE

La performance de la méthode pour la reconnaissance des types de sols proposée dans ce chapitre a été évaluée selon plusieurs méthodes d'apprentissage. Celles-ci ont été réalisées grâce au *workbench* d'apprentissage machine WEKA (Waikato Environment for Knowledge Analysis) (Holmes *et al.*, 1994) qui a été employé en tant que librairie dans une application [18] de type Command Line Interface (CLI) développée en Java. Ainsi, l'évaluation de la performance de la reconnaissance a été déterminée selon la validation croisée en k-plis, par le biais de cette application.

18. https://github.com/FlorentinTh/SoilTypesDetection

La méthode de la validation croisée en k-plis est très largement utilisée dans la littérature de l'apprentissage machine au sens large (Vail et Veloso, 2004; Arlot et Celisse, 2010; Kertesz, 2016). En effet, l'avantage principal de cette technique est qu'elle permet l'utilisation de l'intégralité des données disponibles aussi bien pour la phase d'entraînement que pour la reconnaissance. De plus, celle-ci va permettre de minimiser le biais potentiel d'un ensemble de données de validation construit de manière empirique, c'est-à-dire, réduire les problèmes liés à une mauvaise distribution de ces données comme le sur-apprentissage.

Le fonctionnement de la validation croisée en k-plis est relativement simple. Prenons par exemple un ensemble de données binaires (« $+$ » et « $-$ ») sur lequel on souhaite mettre en place un système de reconnaissance grâce à un algorithme d'apprentissage dont la performance est évaluée avec une validation croisée où $k = 3$. La première étape consiste à découper cet ensemble de données en trois plis mutuellement disjoints. Néanmoins, puisque les données dans l'ensemble initial ne sont pas triées, il est important de s'assurer, lors du découpage, que les instances étiquetées soit par un « $+$ » soit par un « $-$ » soient réparties selon la même fréquence dans les trois plis. C'est le principe de la stratification des données. Dans cet exemple, les trois plis sont notés A, B et C, tel qu'illustré par la Figure 4.8. La deuxième étape de ce processus est d'entraîner une première fois l'algorithme d'apprentissage avec l'ensemble $B \cup C$ et d'évaluer la performance de la reconnaissance avec A. Cette mesure est notée E_A. Ensuite, cette étape est répétée pour les deux plis restant en utilisant les ensembles $A \cup C$ puis $A \cup B$ pour l'entraînement et respectivement B puis C pour quantifier la performance de la

reconnaissance, ce qui permet alors d'obtenir E_B et E_C. Enfin, la performance globale de la reconnaissance (E), est estimée par la moyenne des mesures obtenues pour chaque pli.

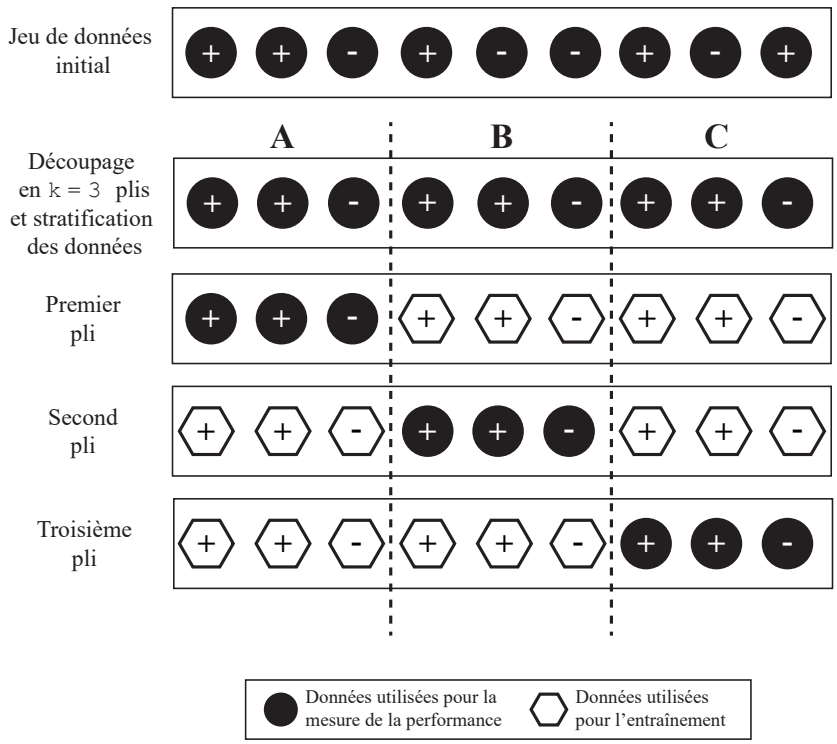


Figure 4.8 : Exemple d'une validation croisée où $k = 3$ plis sur des données binaires.

Depuis l'apparition de l'apprentissage machine, de nombreux travaux ce sont concentrés sur l'optimisation de l'évaluation de la performance de ces systèmes (Witten *et al.*, 2016). Ainsi, le nombre de plis à utiliser dans une validation croisée admis comme standard aujourd'hui est $k = 10$. De ce fait, pour que l'évaluation du système proposé dans ce chapitre demeure cohérente avec celles définies dans la littérature, une validation croisée en 10 plis a donc été retenue.

4.4.3 RÉSULTATS OBTENUS

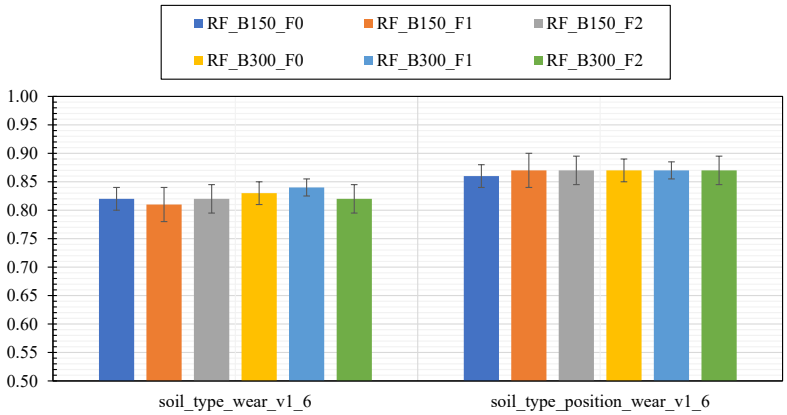
VERSION 1 DU WEARABLE DEVICE

Dans un premier temps, la performance du système de reconnaissance des types de sols a été évaluée sur les données obtenues avec la version 1 du *wearable device*. Pour ce faire, plusieurs modèles d'apprentissage ont été construits en utilisant différents paramètres pour l'algorithme *random forest*. Puisqu'il est préférable qu'une forêt d'arbres décisionnels soit composée d'un grand nombre d'arbres (Breiman, 2001), les deux valeurs : $B = 150$ et $B = 300$ arbres ont été comparées—ceci dans le but de répondre aux exigences proposées par Breiman (2001), ainsi que pour conserver un temps de calcul acceptable. En ce qui concerne le paramètre relatif au nombre de variables aléatoires (F), les trois valeurs préconisées dans la littérature, soit $F_0 = \lfloor \frac{1}{2}\sqrt{m} \rfloor$, $F_1 = \lfloor log_2(m) + 1 \rfloor$ et $F_2 = \lfloor \sqrt{m} \rfloor$ ont également été comparées. Enfin, de par l'implémentation utilisée dans ce système, la fonction employée pour mesurer la qualité de la division des nœuds de l'arbre (C) demeure l'évaluation du gain en information basé sur l'entropie de Shannon, pour chaque expérimentation. La Figure 4.9a montre une représentation graphique des résultats obtenus pour chaque paramétrage de l'algorithme *random forest*, évalué sur les deux ensembles de données.

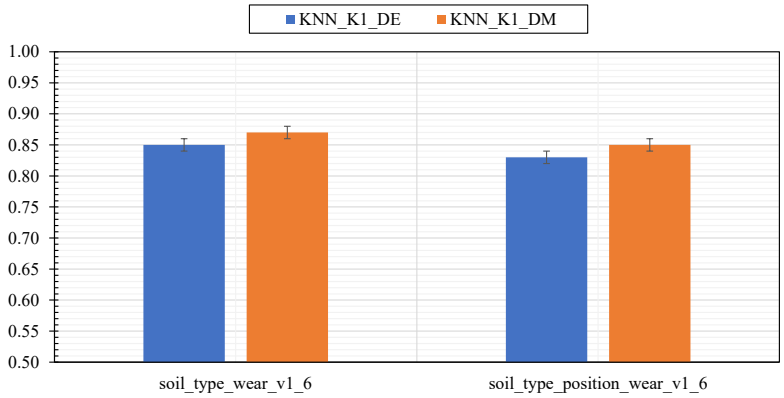
De la même manière, la performance de l'algorithme des k plus proches voisins a été évaluée afin que les résultats puissent être comparés avec ceux obtenus pour l'algorithme *random forest*. Pour ce faire, les mesures de la distance euclidienne (D_e) ainsi que de la distance de Manhattan (D_m) ont toutes deux été utilisées. Afin de déterminer le nombre de

voisins (k) optimal pour ce système de reconnaissance, toutes les valeurs comprises entre $k = 1$ et $k = \sqrt{m}$, où m correspond au nombre de caractéristiques discriminantes, ont été essayées et les résultats obtenus ont été comparés de manière empirique. Ainsi, il est apparu que le nombre approprié de voisins à considérer était d'un seul. La Figure 4.9b montre une représentation graphique des résultats obtenus pour chaque paramétrage de l'algorithme k-NN, évalué grâce à une recherche linéaire du plus proche voisin pour les deux jeux de données. Par ailleurs, l'ensemble des résultats de cette première expérimentation est fourni en détail, pour les deux algorithmes, dans l'Annexe A.1.

En s'appuyant sur ces résultats, il est possible d'affirmer que les paramètres optimaux qui sont suggérés pour la méthode de reconnaissance des types de sols, lorsqu'elle est réalisée avec l'algorithme *random forest* sont : $B = 300$ arbres et un nombre de variables aléatoires équivalent à $F_1 = \log_2(m) + 1$, où m correspond au nombre de caractéristiques discriminantes. En effet, c'est avec ces paramètres que les meilleurs résultats ont été observés, c'est-à-dire, une *F-mesure* moyenne des deux jeux de données de 86%. De plus, la même *F-mesure* moyenne a été obtenue avec l'algorithme k-NN lorsque celui-ci est configuré avec la fonction de distance de Manhattan. En comparaison, l'emploi de la fonction de distance euclidienne a, quant à elle, montré de moins bons résultats. De ce fait, la mesure de distance qui semble la plus optimale pour cette algorithme est donc D_m.



(a) Les *F-mesures* obtenues sur les deux ensembles de données avec les différentes configurations pour l'algorithme *random forest*.



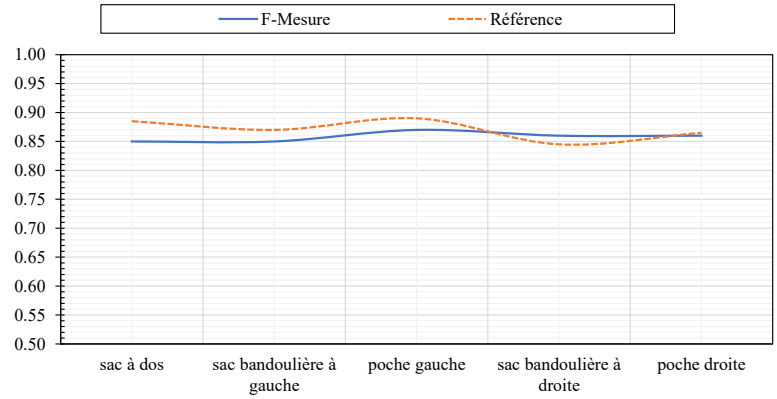
(b) Les *F-mesures* obtenues sur les deux ensembles de données avec avec les différentes configurations pour l'algorithme des *k* plus proches voisins.

Figure 4.9 : Les *F-mesures* obtenues sur les deux ensembles de données avec les différentes configurations pour les algorithmes *random forest* et *k*-NN avec la version 1 du *wearable device* (Thullier *et al.*, 2017).

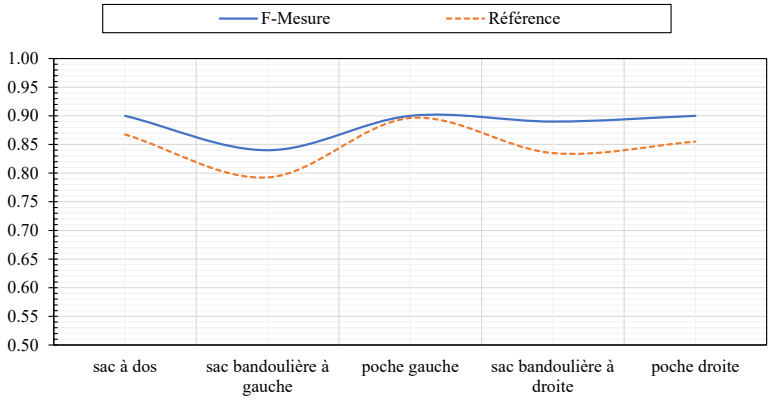
Afin de s'assurer que la reconnaissance des sols soit fonctionnelle, et ce, indépendamment de la position choisie par l'utilisateur pour le porter, cinq jeux de données distincts ont été construits à partir des ensembles de données initiaux utilisés dans l'expérience pré-

cédente. Ainsi, ils constituent un ensemble de données par emplacement dont les étiquettes correspondent aux types de sols uniquement. Ensuite, l'évaluation de la performance de reconnaissance a été réalisée avec les deux mêmes algorithmes d'apprentissage, configurés avec les paramètres qui ont été déterminés optimaux dans l'expérimentation précédente. Les résultats obtenus sont données par les courbes continues représentées sur la Figure 4.10. De plus, afin de déterminer si la position du *wearable device* avait un impact significatif sur le taux de reconnaissance, la moyenne des *F-mesures* obtenues pour chaque position par rapport aux types de sols a été évaluée à l'aide du jeu de données « `soil_type_position_wear_v1_6` ». Ce sont les courbes discontinues également représentés sur la Figure 4.10. Celles-ci déterminent alors les valeurs de référence pour chaque position.

L'expression graphique donnée par ses différentes courbes permet de constater qu'aucune position distincte du *wearable device* n'a un impact significatif sur la performance de la reconnaissance des types de sols. En effet, les différences maximum observées dans les *F-mesures* sont de 2% et 6%, respectivement avec l'algorithme *random forest* et *k*-NN. De plus, il est possible, pour chaque algorithme, d'observer une similarité entre la tendance de la courbe exprimant les résultats et la courbe de référence.



(a) Les *F-mesures* obtenues pour chaque position lors de l'évaluation de l'indépendance de positionnement avec l'algorithme *random forest*, où $B = 300$ et $F = F_1$.



(b) Les *F-mesures* obtenues pour chaque position lors de l'évaluation de l'indépendance de positionnement avec l'algorithme k-NN où la mesure de distance utilisée est D_m.

Figure 4.10 : Les *F-mesures* pour chaque position par rapport aux valeurs références permettant l'évaluation de l'indépendance de positionnement du *wearable device* dans sa version 1, pour les deux algorithmes : *random forest* et k-NN (Thullier *et al.*, 2017).

VERSION 2 DU WEARABLE DEVICE

Dans un second temps, les deux expérimentations précédentes ont été reproduites selon la même procédure avec la version 2 du *wearable device*. Puisque cette version offre un plus grand nombre d'axes pour la centrale inertielle, l'objectif était de vérifier l'impact des axes supplémentaires sur la reconnaissance des types de sols afin de proposer l'implémentation matérielle la plus précise possible pour le *wearable device*. La Figure 4.11 présente les résultats pour les deux algorithmes configurés avec les paramètres optimaux identifiés précédemment pour les deux jeux de données différents. Pour atténuer la perte d'information dans cette expérimentation comparativement à la première, les données de l'IMU avec seulement 6 axes ont également été traitées.

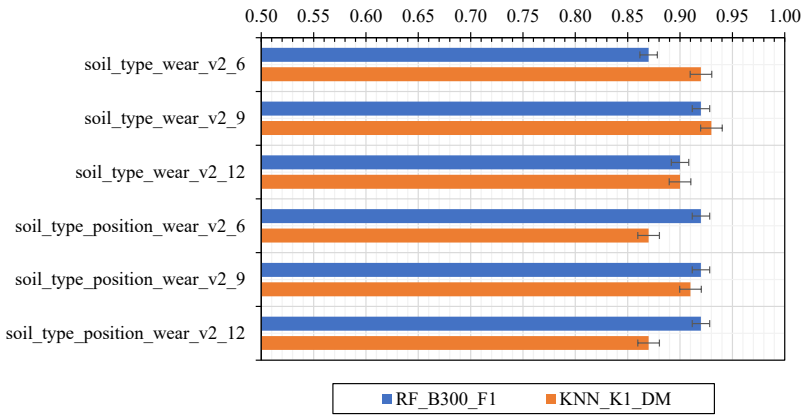
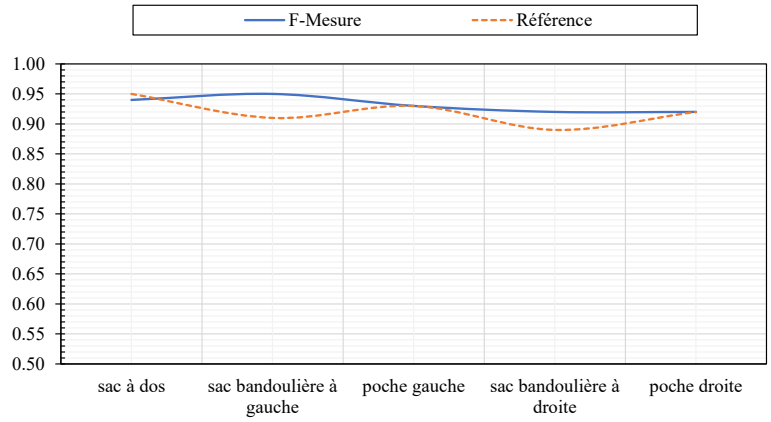


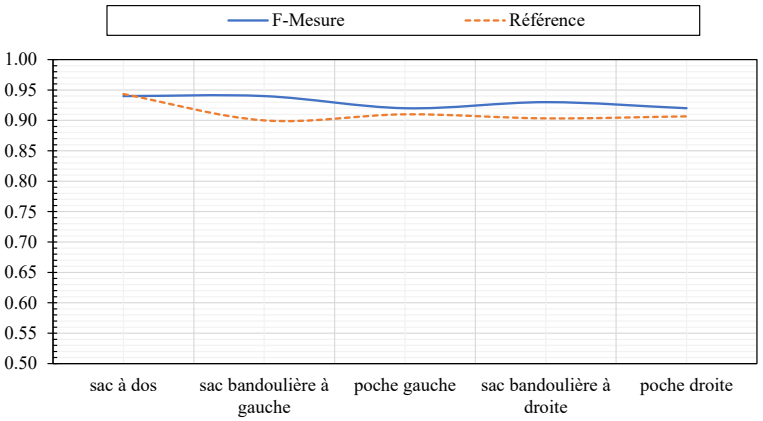
Figure 4.11 : Les *F-mesures* obtenues sur les deux ensembles de données enregistrés avec la version 2 du *wearable device* pour les algorithmes *random forest* et *k*-NN configurés avec les paramètres optimaux (Thullier *et al.*, 2017).

Premièrement, ces résultats montrent que la perte d'information entre les expériences réalisées avec la première version du dispositif et la seconde n'ont pas un impact significatif sur la reconnaissance des types de sols. En effet, la même *F-mesure* a été obtenue avec l'algorithme *random forest*. Par ailleurs, bien que la *F-mesure* obtenue pour l'algorithme des *k* plus proches voisins ait augmentée de 6%, ce résultat était attendu principalement en raison du fonctionnement même de l'algorithme. De plus, le `LSM9DS1` est, en théorie, une centrale inertielle beaucoup plus précise que celle utilisée lors de l'expérimentation avec la version 1 du dispositif. Par ailleurs, une amélioration de la reconnaissance a été constatée avec le jeu de données produit par l'IMU 9-axes, tandis qu'une amélioration négligeable a été observée avec l'ensemble de données contenant les angles d'Euler. Enfin, les résultats détaillés fournis en Annexe A.2, permettent de confirmer que les paramètres utilisés dans la configuration des deux algorithmes pour cette expérimentation sont toujours les plus optimaux.

En ce qui concerne l'indépendance du positionnement du *wearble device* dans sa seconde version, les mêmes observations peuvent être établies selon les résultats qui ont été obtenus. Ils sont exposés en Figure 4.12. En effet, les mêmes similarités sont visibles entre les tendances des courbes exprimant les résultats et celles de référence.



(a) Les *F-mesures* obtenues pour l'évaluation de l'indépendance de positionnement avec l'algorithme *random forest* configurés avec les paramètres optimaux.



(b) Les *F-mesures* obtenues pour l'évaluation de l'indépendance de positionnement avec l'algorithme *k*-NN configurés avec les paramètres optimaux.

Figure 4.12 : Les *F-mesures* obtenues par rapport aux valeurs de référence permettant l'évaluation de l'indépendance de positionnement du *wearable device* dans sa version 2, pour les deux algorithmes : *random forest* et *k*-NN configurés avec les paramètres optimaux (Thullier *et al.*, 2017).

TÉLÉPHONE INTELLIGENT

En dernier lieu, l'analyse de la performance de la reconnaissance des types de sols a été reproduite selon la même procédure, en utilisant un téléphone intelligent. Les résultats obtenus avec les paramètres optimaux pour les deux algorithmes sont illustrés en Figure 4.13. Néanmoins, l'Annexe A.3 présente de façon plus précise l'ensemble des résultats acquis pour tous les jeux de données et toutes les configurations possibles pour les algorithmes de reconnaissance. Ces résultats montrent une excellente performance de reconnaissance globale. Bien que ces résultats demeurent principalement comparables à ceux obtenus avec le *wearable device*, le taux de reconnaissance, dans cette configuration, ne semble pas affecté ni par le nombre d'axes admis par la centrale inertielle, ni par l'algorithme de reconnaissance et ses paramètres.

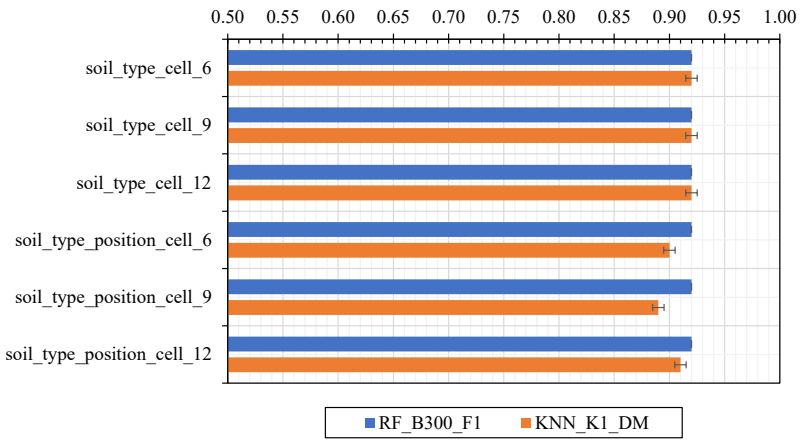


Figure 4.13 : Les *F-mesures* obtenues sur les deux ensembles de données enregistrés avec le téléphone intelligent pour les algorithmes *random forest* et k-NN configurés avec les paramètres optimaux (Thullier *et al.*, 2017).

4.4.4 DISCUSSION DES RÉSULTATS OBTENUS

Dans la section précédente, l'ensemble des résultats ont été présentés sous forme de synthèse. Ceux-ci se sont révélés particulièrement précis dans la globalité des expérimentations qui ont été menées, sachant que les données inertielles sont parmi les données qui demeurent les plus sensibles au bruit. En effet, avec la première version du *wearable device*, une performance de 86% a été obtenue en meilleur cas. Par ailleurs, il a été constaté que la version améliorée du dispositif (version 2) a permis une amélioration de 6% des résultats de la reconnaissance des sols—soit une performance globale de 92% en meilleur cas et en utilisant une centrale inertielle à 9-axes au lieu de 6-axes. Néanmoins, considérant le temps supplémentaire requis pour le traitement des angles d'Euler et la plus faible performance de reconnaissance obtenue dans cette configuration, il est possible d'affirmer que l'utilisation d'un IMU 9-axes demeure la configuration matérielle idéale pour réaliser la reconnaissance des types de sols. Une telle différence entre les deux versions des dispositifs peut s'expliquer simplement par le fait que l'IMU 6-axes directement intégré dans l'*Arduino 101* est de moins bonne qualité et offre une moins bonne précision que le `LSM9DS1`.

Par ailleurs, les résultats obtenus avec le téléphone intelligent ont démontré une légère amélioration de la performance de reconnaissance globale. De plus, cette expérimentation a dévoilé une certaine stabilité dans les résultats obtenus malgré les différentes configurations qui ont été appliquées. L'hypothèse qui est formulée pour expliquer cet effet questionne la mise en place automatique d'un filtre sur les données issues du capteur inertiel (*p. ex.* un filtre passe-bas). Il est en effet possible que le système d'exploitation applique un tel filtre avant

que les données soient transmises à notre application. Par conséquent, cela implique que les données exploitées dans cette expérimentation ne sont pas exactement les données brutes et donc, qu'elles incluent beaucoup moins de bruit que les données produites par notre dispositif.

Par ailleurs, en ce qui concerne l'hypothèse de l'indépendance de la position du *wearable device*, les comparaisons qui ont été effectuées ont montré que la position du dispositif n'impactait pas la performance de la reconnaissance des sols. Cependant, puisque les expérimentations ont impliqué un nombre relativement réduit de personnes, il apparaît pertinent d'en réaliser de nouvelles incluant un nombre beaucoup plus important de participants dans le but de renforcer cette affirmation.

Finalement, bien que l'ensemble des résultats obtenus soient très satisfaisants, nous pensons qu'il est encore possible de les améliorer. En effet, il serait envisageable de proposer une étape de prétraitement des données brutes afin d'en affiner la qualité. Cette étape pourrait, par exemple, constituer l'application de techniques telles qu'un filtre passe-bas ou un filtre de Kalman. De plus, il est possible que certaines caractéristiques proposées dans notre système ne soient pas suffisamment discriminantes et pourraient donc être supprimées. Par conséquent, cela permettrait principalement de réduire la dimension de nos ensembles de données. Ces deux possibilités d'amélioration pour notre système peuvent donc potentiellement aboutir à une meilleure performance de la reconnaissance des types de sols.

4.5 CONCLUSION

Dans ce chapitre, une nouvelle méthode pour reconnaitre les types de sols grâce aux données inertielles produites par la démarche humaine a été introduite. Pour ce faire, un *wearable device* a été conçu dans le but d'enregistrer les données inertielles générées par son porteur. Dans une première version, ces données ont été obtenues par le biais d'un IMU 6-axes déjà intégré au système électronique sur lequel notre choix s'est porté, c'est-à-dire, un *Arduino 101*. Dans une seconde version, un IMU 9-axes, plus précis, a été utilisé. L'enregistrement des données a été effectué grâce à une application *Android*. Cette dernière était en charge de récupérer et d'étiqueter les données inertielles *via* une connexion Bluetooth préalablement établie.

L'évaluation de cette méthode a été réalisée selon plusieurs expérimentations, toutes basées sur plusieurs étapes. La première concerne l'extraction de caractéristiques temporelles et fréquentielles sur les données brutes récoltées. Ensuite, les deux algorithmes d'apprentissage machine *random forest* et des k plus proches voisins ont été comparés. Enfin, la performance de la reconnaissance a pu être évaluée dans une dernière étape grâce au calcul de la *F-mesure* réalisée par le biais de la méthode de la validation croisée en 10-plis. La première expérimentation s'est déroulée avec l'implication de neuf participants à qui il a été demandé de marché sur quatre différents types de sols (`ciment`, `gravier`, `neige` et `sable`) en portant le *wearable device* selon cinq positions distinctes. La seconde expérimentation a repris les mêmes grandes lignes du protocole de la première expérimentation à l'exception faite qu'un téléphone intelligent a été ajouté en plus du dispositif pour l'enregistrement des données

inertielles. De plus, puisque celle-ci s'est déroulée en été, il a été impossible d'établir une récolte de données sur la neige. Aussi, certains participants ayant quitté le laboratoire entre temps, cette expérimentation n'a impliquée que six participants parmi les neuf qui étaient présents lors de la première.

Les résultats obtenus grâce à ces expérimentations ont, dans un premier temps, permis de déterminer les paramètres optimaux à utiliser dans la configuration des algorithmes d'apprentissage machine. De plus, les différentes évaluations de la performance de reconnaissance ont montré une excellente précision, soit 86% au mieux pour la première expérimentation et 92% au mieux pour la seconde. Cette amélioration a permis de déterminer qu'une centrale inertielle à 9-axes était la configuration matérielle à adopter pour proposer une reconnaissance la plus précise possible. Enfin, bien que ces expérimentations aient également permis de statuer que la position du *wearable device* n'avait aucun impact sur la performance de la reconnaissance des sols, il a été identifié que cette affirmation doit être renforcée par d'autres analyses incluant un nombre plus important de participants.

CHAPITRE V

UNE ARCHITECTURE D'HABITATS INTELLIGENTS DISTRIBUÉE

Grâce aux chapitres précédents, il a été vu que les *wearable devices* sont de plus en plus utilisés dans le processus de reconnaissance d'activités au sein des habitats intelligents. Aussi, ces dispositifs ont également été employés dans l'objectif de répondre à plusieurs autres problématiques concrètes, comme la méthode de reconnaissance des types de sols qui été présentée au Chapitre 4, qui sont relatives à l'assistance des résidents de ces habitats, au sens large. Néanmoins, l'engouement croissant pour ces dispositifs a permis d'identifier plusieurs problématiques quant à leur utilisation dans ce contexte de recherche particulier. En effet, nous avons été en mesure d'identifier que la plupart des différentes architectures d'habitats intelligents qui ont été proposées antérieurement n'ont pas été développées dans une optique évolutive. Lorsqu'une intégration matérielle des *wearable device* s'est montré facilement réalisable, comme dans l'architecture CASAS (Cook *et al.*, 2013), cela était au détriment de la fiabilité du processus de la reconnaissance d'activités principalement affectée par une conception d'habitat admettant plusieurs points de défaillance. Par conséquent, la principale question de recherche qui est traitée dans ce chapitre est : *« comment faire évoluer les architectures de maisons intelligentes pour leur permettre de mieux s'adapter aux divers types de capteurs (ambiants et wearable devices) tout en garantissant un excellent niveau de fiabilité dans l'accomplissement des différents processus d'apprentissage ? »*.

Pour répondre à cette question, le travail présenté dans ce chapitre propose un type différent d'architecture de maisons intelligentes. Cette implémentation a été développée en s'inspirant des architectures de *cloud* privées sur site qui disposent de nombreux mécanismes qui permettent de s'assurer que l'infrastructure matérielle demeure hautement disponible, ceci dans le but de garantir à la fois un excellent niveau de fiabilité, mais également une évolutivité notoire. Ainsi, cette architecture vise à résoudre la plupart des problèmes identifiés au sein des habitats précédemment proposés sans qu'il soit nécessaire de remplacer leur structure matérielle dans leur intégralité. En effet, l'architecture présentée dans ce chapitre a été pensée pour à la fois être déployée en l'état, ou pour s'intégrer à celles existantes, remplaçant ainsi les entités centrales des architectures monolithiques.

La suite de ce chapitre comporte une première section qui présente en détail l'architecture d'habitats intelligents proposée. Ensuite, les expérimentations sont décrites dans une seconde section. Finalement, ce chapitre propose une discussion des observations réalisées et dresse une conclusion quant à ce second travail, dans une dernière partie.

5.1 ARCHITECTURE PROPOSÉE

Bien que l'architecture introduite dans ce chapitre ait été conçue pour être compatible avec la majorité des architectures de maisons intelligentes présentes dans la littérature, celle-ci peut surtout être considérée comme une évolution du travail proposé par Plantevin *et al.* (2018). En effet, puisque les auteurs ont concentré leurs efforts sur une intégration matérielle uniforme des transducteurs (capteurs et effecteurs), ils ont amorcé les mêmes perspectives

d'amélioration des habitats intelligents existants en se penchant principalement sur les capteurs ambiants. Par conséquent, cette thèse ne prendra pas en considération ce type de capteurs, mais va plutôt traiter exclusivement des *wearable devices*. De plus, l'objectif principal de ce travail est d'aller plus loin dans l'idée de rendre ces architectures plus fiables, mais aussi plus flexibles et évolutives. Pour ce faire, ce chapitre présente une implémentation qui repose sur l'utilisation des microservices plutôt qu'une approche monolithique.

5.1.1 MICROSERVICES

Les architectures de microservices sont récemment apparues comme un nouveau paradigme pour la programmation d'applications (Dragoni *et al.*, 2017) et dont l'origine se base sur le concept d'Architecture Orientée Services (AOS) (MacKenzie *et al.*, 2006). Comme le montre la Figure 5.1, cette approche suggère de diviser une application en un ensemble de services plus petits, indépendants et interconnectés. En comparaison aux applications monolithiques, où tous les composants logiciels se trouvent dans une seule instance, les services exécutent des fonctions plus détaillées et précises. Bien que les architectures monolithiques soient simples à implémenter, les architectures de microservices présentent, elles aussi, différents avantages. La première concerne l'*agilité* des microservices. En effet, puisque les applications sont fragmentées au niveau le plus élémentaire de leurs fonctionnalités, chaque service devient individuellement plus facile à maintenir et beaucoup plus rapide à développer et à déployer. Le second avantage principal demeure la *réutilisabilité*. Les microservices permettent aux développeurs de créer des applications en utilisant certains fragments déjà

existants. De plus, la barrière entre les technologies (*p. ex.* les langages de programmation) est réduite puisque les services fonctionnent de manière indépendante. En ce sens, cet *agnosticisme technologique* permet d'obtenir des applications hétérogènes du point de vue des technologies dans leur conception. Par ailleurs, un autre avantage offert par les architectures

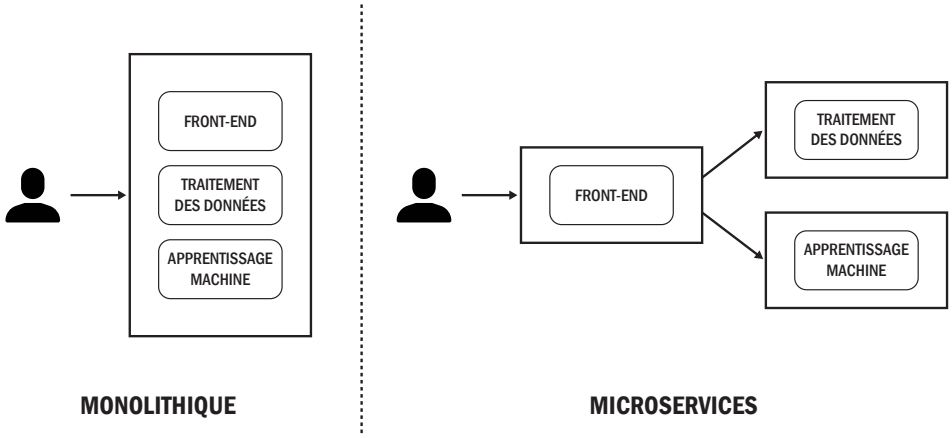


Figure 5.1 : Comparaison entre les architectures monolithiques et de microservices selon un exemple d'application qui vise à résoudre un problème d'apprentissage machine par le biais d'une interface graphique.

de microservices concerne la mise à l'échelle des différents composants à mesure que la demande pour une application augmente. En cas de forte demande ponctuelle, il est possible soit d'augmenter la quantité de ressources allouées, soit d'augmenter le nombre d'instances pour les services les plus impactés, plutôt que pour l'intégralité de l'application. Il s'agit respectivement de l'*extensibilité dynamique horizontale* et de l'*extensibilité dynamique verticale*. Enfin, puisque les composants d'une application dans une architecture de microservices sont généralement distribués, l'application est en mesure de supporter les pannes au niveau des services qui la composent. En effet, lorsqu'un service est en panne, la fonctionnalité dont il

est responsable devient inopérante, mais le fonctionnement du reste de l'application n'est pas altéré. De plus, il est possible de mettre en place différents mécanismes pour remédier aux défaillances inattendues afin d'augmenter la résilience et la *fiabilité* de l'ensemble du système.

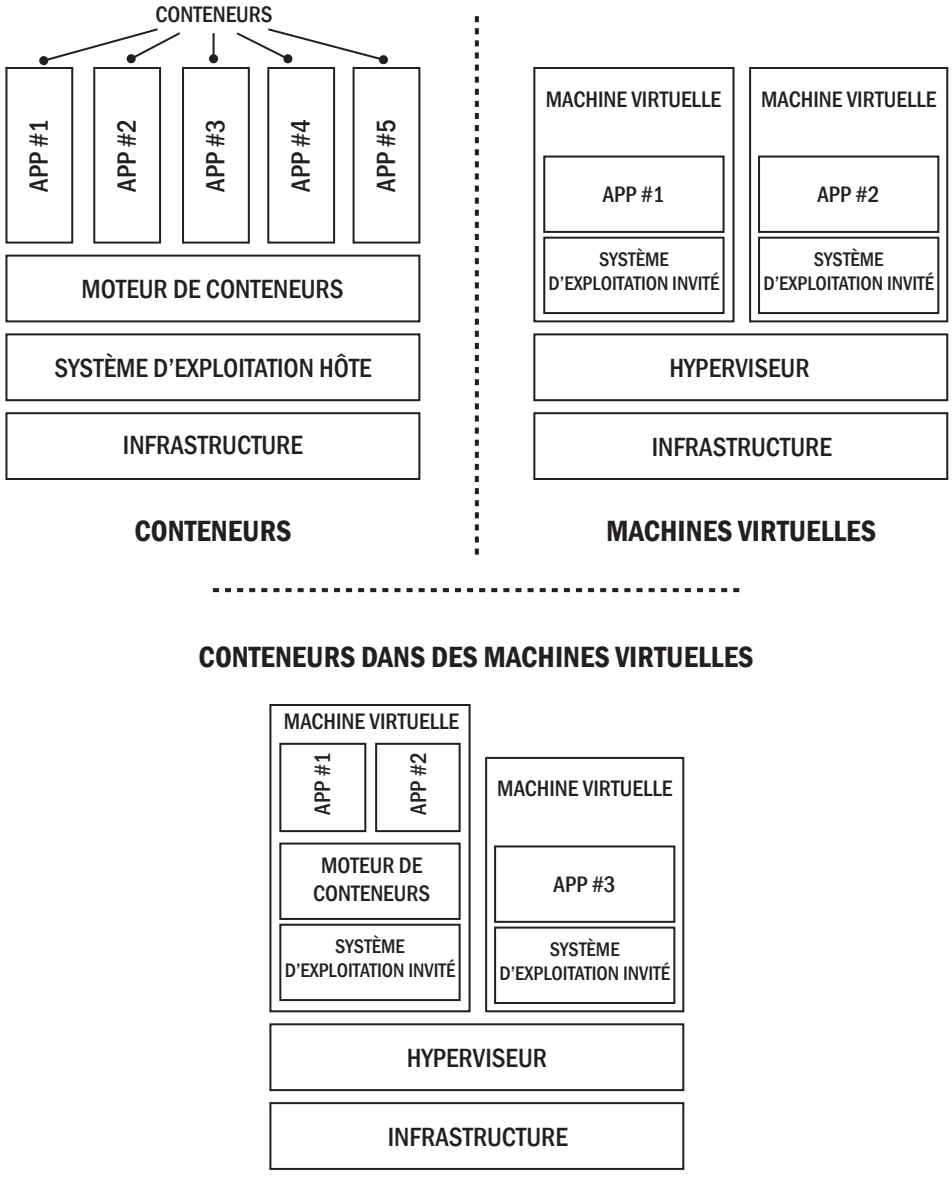


Figure 5.2 : Les trois principales techniques de virtualisation pour la mise en place d'une architecture de microservices.

Actuellement, il existe de nombreuses façons de mettre en place une architecture de microservices, la plus courante étant l'utilisation d'outils de virtualisation tels que les machines virtuelles (VMs) traditionnelles, les conteneurs ou les conteneurs à l'intérieur de VMs. La Figure 5.2 fournit une représentation graphique de ces trois différentes implémentations. Même si le choix de la méthode de virtualisation n'affecte pas les possibilités offertes par les architectures de microservices, il existe des différences significatives entre l'utilisation de VMs traditionnelles et les conteneurs qu'il semble important de détailler. La principale différence réside dans le fait que les conteneurs proposent une virtualisation au niveau du système d'exploitation alors que les VMs offrent une virtualisation au niveau du matériel. En ce qui concerne les VMs, l'hyperviseur permet de cloisonner des portions du matériel. En effet, son rôle est d'attribuer à chaque machine virtuelle les ressources nécessaires en fonction des ressources physiques du système hôte. De manière générale, il existe deux types d'hyperviseurs. Dans un premier temps, il y a les hyperviseurs de système natif qui s'exécutent directement sur le matériel du système hôte. En ce sens, ceux-ci prennent la place du système d'exploitation de l'hôte et planifient directement l'utilisation des ressources allouées aux machines virtuelles en fonction du matériel. Dans un second temps, les hyperviseurs hébergés fonctionnent comme une couche logicielle supplémentaire au sein du système d'exploitation hôte. Ainsi, ils permettent de dissocier les systèmes d'exploitation invités (virtualisés) du système d'exploitation hôte.

Alternativement, les conteneurs permettent aux instances virtuelles de partager un système d'exploitation hôte unique. Par conséquent, puisque les conteneurs n'ont pas à

embarquer un système d'exploitation, les images virtualisées demeurent beaucoup plus légères que celles des VMs traditionnelles. De plus, les conteneurs sont, au même titre que les VMs, isolés du système hôte, c'est-à-dire, qu'ils s'exécutent dans des espaces séparés à la fois les uns des autres, mais également de certaines parties du système d'exploitation hôte. Une telle isolation permet donc une utilisation plus efficace des ressources. Enfin, les conteneurs sont très rapides à créer et à détruire puisqu'il n'est pas nécessaire de démarrer ou d'arrêter un système d'exploitation à chaque fois. En effet, les conteneurs s'occupent uniquement de compléter le processus dont ils sont en charge.

En somme, une VM est une émulation d'un système informatique tandis que les conteneurs sont toutes les unités logicielles qui regroupent le code et toutes ses dépendances requises (*p. ex.* les librairies ou les binaires) qui composent une application. De ce fait, celle-ci peut s'exécuter de façon fiable et rapide d'un environnement à un autre. Ces deux technologies ne sont pas incompatibles puisqu'il est possible de les utiliser ensemble. Cependant, l'exécution de conteneurs à l'intérieur de machines virtuelles est, la plupart du temps, le résultat de l'évolution d'environnements de virtualisation matures déjà existants. Généralement, les conteneurs sont exécutés directement sur le système hôte afin d'optimiser les performances et la latence ou de réduire les coûts des licences des outils de virtualisation et des systèmes d'exploitation.

5.1.2 ORGANISATION MATÉRIELLE

L'organisation matérielle de l'implémentation de l'architecture d'habitats intelligents présentée dans ce chapitre est illustrée par la Figure 5.3. Afin de supporter l'architecture de

microservices, cette organisation repose sur un ensemble de cinq nœuds principaux (`m0`, `m1`, `m2`, `w0` et `w1`), puisque ceci représente le nombre minimum de nœuds nécessaires pour former un *cluster* distribué fiable. Par ailleurs, l'implémentation proposée suggère l'utilisation de deux nœuds supplémentaires (`f0` et `f1`) sur chacun desquels une unité de stockage de type RAID (Redundant Array of Independent Disks) est interfacée. L'objectif de ces nœuds est de fournir un système de fichiers distribué à l'ensemble du *cluster*. En effet, puisque certains conteneurs ont la capacité de monter différents volumes sur le système hôte, un tel système de fichiers offre un moyen fiable de partager des données entre chaque conteneur, quel que soit le nœud sur lequel ils s'exécutent. En outre, tous les nœuds qui composent le *cluster* et le système de fichiers distribué sont connectés à un commutateur réseau pour leur permettre de communiquer entre eux. De plus, l'ensemble de l'architecture est isolé à l'intérieur d'un réseau local afin de prévenir les problèmes de sécurité qui pourraient compromettre les expériences

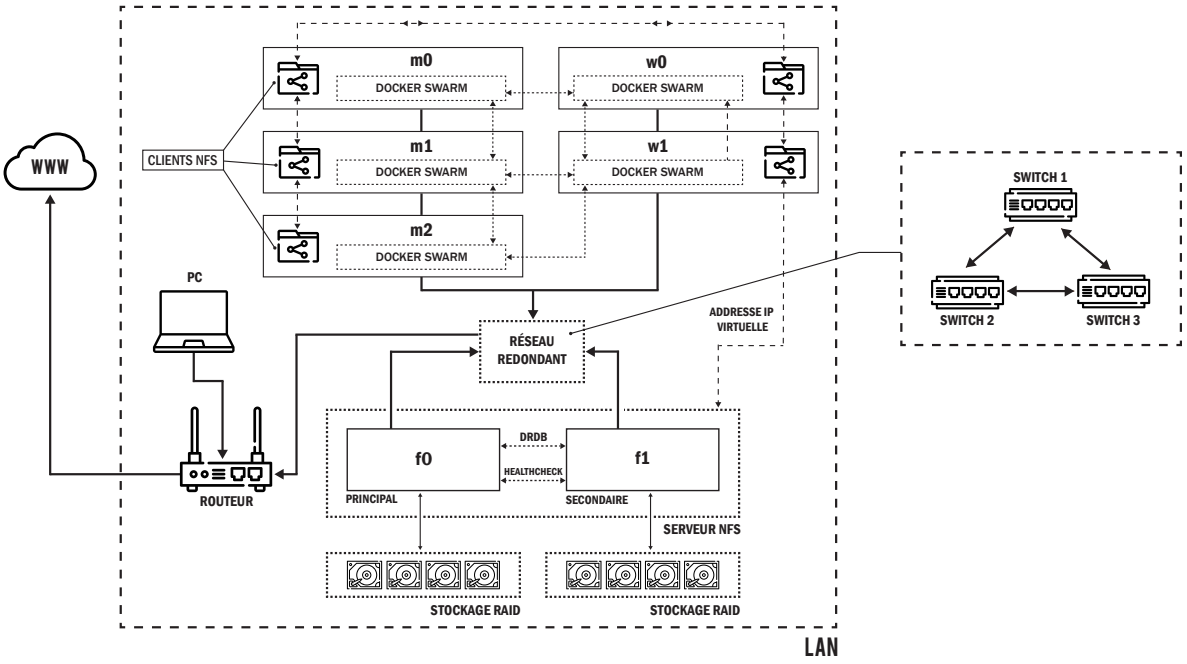


Figure 5.3 : Organisation matérielle de l'architecture d'habitats intelligents proposée illustrant l'implémentation d'un *cluster* pour le support de l'architecture de microservices ainsi que le système de fichiers distribué.

et de préserver l'anonymat des participants. Toutefois, pour faciliter la configuration et la maintenance de l'architecture, cette dernière demeure accessible de manière ad hoc avec un ordinateur *via* un routeur qui est, quant a lui, connecté à la fois au Web et au réseau local.

5.1.3 SYSTÈME DE FICHIERS DISTRIBUÉ

Le système de fichiers distribué fourni par les nœuds `f0` et `f1` repose sur le protocole Network File System (NFS). Le principe de ce protocole reste simple, car il est implémenté selon le modèle client/serveur. Un serveur NFS gère l'authentification, l'autorisation, l'administration des clients ainsi que les données. Ainsi, une fois autorisés, les clients NFS peuvent

accéder aux données distantes de la même manière qu'ils le font localement. En ce sens, chaque nœud qui compose le *cluster* est autorisé à accéder (lecture) et à manipuler les fichiers (création, mise à jour, suppression) de manière classique. Néanmoins, tous les autres nœuds seront informés des changements qui en résultent.

Cependant, puisque l'idée est de proposer une architecture fiable, le fait de n'avoir qu'un seul serveur NFS ne satisfaisait pas cette contrainte. En ce sens, l'outil DRDB [19] (Distributed Replicated Block Device) a été mis en place afin que les nœuds `f0` et `f1` puissent proposer un système de fichiers à la fois distribué, mais aussi répliqué. DRDB a été paramétré pour effectuer une réplication complète des données stockées entre le serveur principal (`f0`) et le serveur secondaire (`f1`), soit le même fonctionnement qu'un système de stockage de type RAID 1 en réseau. Afin de favoriser l'intégrité des données plutôt que la vitesse de réplication, la mise en place d'une réplication synchrone constante entre le nœud principal et le nœud secondaire a été adoptée plutôt qu'une réplication asynchrone ou semi-synchrone. Ainsi, les opérations de manipulation des fichiers réalisées sur le nœud principal ne sont considérées comme terminées que lorsque les écritures sur le disque local et sur le disque distant du nœud secondaire sont confirmées par ce dernier. Néanmoins, DRDB ne fournit aucune fonctionnalité relative à un quelconque mécanisme garantissant la fiabilité du système de fichiers en cas de panne d'un des deux nœuds. En ce sens, l'outil Heartbeat [20] a également été adopté dans cette implémentation. Cet outil permet de proposer un mécanisme permettant de garantir

19. https://www.linbit.com/drbd/

20. http://www.linux-ha.org/doc/man-pages/re-heartbeat.html

la haute disponibilité du système de fichiers distribué. En tant que processus d'arrière-plan, le logiciel vise à surveiller l'état des deux serveurs DRDB. En cas de défaillance du nœud principal (`f0`), l'adresse IP virtuelle (VIP) du système de fichiers distribué qui pointe vers `f0` est automatiquement réaffectée pour identifier le nœud secondaire (`f1`). Ainsi, l'accès aux données n'est en aucun cas compromis lors de la perte d'un seul nœud, puisque cette opération est effectuée de manière transparente pour les clients NFS.

5.1.4 ORCHESTRATION DES CONTENEURS

L'implémentation de l'architecture de microservices proposée dans ce chapitre repose d'abord sur le moteur de conteneurs Docker [21], qui a été installé sur chaque nœud qui compose le *cluster* (`m0`, `m1`, `m2`, `w0` et `w1`). De plus, afin de pouvoir mettre en place le *cluster* distribué grâce à l'ensemble de ces nœuds, il était nécessaire d'utiliser un orchestrateur. L'orchestration de conteneurs fait référence à un ensemble de techniques qui permettent de gérer les différents cycles de vie des conteneurs ainsi que leurs environnements dynamiques. Plus précisément, les outils d'orchestration permettent d'automatiser plusieurs tâches telles que la configuration, l'ordonnancement, l'approvisionnement et la mise à l'échelle des conteneurs, mais également la répartition de charge, l'allocation des ressources et la sécurité des interactions entre les conteneurs. Il existe actuellement plusieurs outils d'orchestration de conteneurs populaires

21. https://www.docker.com

tels que Kubernetes [22], Apache Mesos [23] et Docker Swarm [24]. Pour notre architecture, c'est l'outil Docker Swarm qui a été adopté. Plusieurs facteurs ont motivé ce choix. Tout d'abord, cet outil est plus simple à mettre en place puisqu'il est conditionné, distribué et totalement intégré au moteur de conteneur qui est déjà utilisé. En outre, bien qu'il est considéré comme moins extensible parmi les trois outils mentionnés, celui-ci offre, tel que détaillé dans la suite de cette section, tous les mécanismes requis pour créer et gérer le *cluster* de cette architecture.

De manière générale, un *cluster* Docker Swarm est composé de plusieurs nœuds sur lesquels le moteur Docker est installé. Ces nœuds peuvent adopter deux types de rôles—un rôle de gestionnaire (*manager*) ou un rôle de travailleur (*worker*). Les nœuds de type *manager* sont ceux qui reçoivent les fichiers de définition des services à exécuter au sein du *cluster*. De plus, les *managers* distribuent les tâches qui composent ces services aux nœuds de type *worker*, en fonction de leurs fichiers de définition. Ainsi, ce sont les nœuds *manager* qui sont responsables de l'orchestration et de la gestion du *cluster*. Dans ce contexte, les services font référence à un ensemble de tâches, c'est-à-dire des conteneurs, qui doivent s'exécuter sur les nœuds travailleurs du *cluster* pour réaliser un processus donné. Néanmoins, il est tout à fait possible de définir une politique de placement manuellement. Ainsi, un service peut être soit global, soit répliqué. Lorsqu'un service est global, une unique instance est déployée sur chaque nœud du *cluster* qui est disponible, et ce, sans tenir compte de son rôle. À l'inverse,

22. https://kubernetes.io

23. https://mesos.apache.org

24. https://docs.docker.com/engine/swarm/

lorsqu'un service est répliqué, il est nécessaire de préciser, dans le fichier de définition, le nombre de répliques qui doivent être déployées ainsi que le type de nœuds sur lesquelles elles seront exécutées. Un des nœuds *manager* est alors en charge d'établir cette répartition. Par ailleurs, afin que les nœuds *manager* puissent suivre l'évolution de l'exécution des services, les nœuds travailleurs incluent un agent dont le rôle est de rendre compte, aux *managers*, de l'état des services qui lui sont attribués. Par défaut, les nœuds *manager* sont également des nœuds travailleurs, mais il est possible de les configurer de manière à ce qu'ils n'acceptent aucune charge de travail en plus de leurs tâches de gestion du *cluster*.

En ce qui concerne les *managers*, puisqu'ils représentent les nœuds les plus importants pour la stabilité du *cluster*, il est nécessaire que ce dernier en comporte plusieurs. Plus précisément, préférablement un nombre impair, selon les recommandations préconisées par Docker Swarm. En effet, parmi tous les nœuds de type *manager*, l'un d'entre eux est élu en tant que leader par tous les autres *managers* selon une implémentation de l'algorithme de consensus Raft (MacKenzie *et al.*, 2006). Ainsi, ce nœud particulier est responsable de toutes les décisions pour l'ensemble du groupe de *managers*. Bien qu'il n'y ait pas de limite quant au nombre de nœuds gestionnaires que peut admettre un *cluster* Docker Swarm, la décision concernant l'effectif à mettre en place reste un compromis entre la performance et la tolérance aux pannes. En effet, si le nœud élu comme leader du *cluster* devient indisponible, n'importe quel autre *manager* peut obtenir ce rôle si le quorum Raft, c'est-à-dire, un accord majoritaire entre les nœuds gestionnaires, est possible. Plus précisément, l'algorithme Raft tolère jusqu'à $\lfloor (N-1)/2 \rfloor$ défaillances et exige un quorum de $\lfloor (N/2)+1 \rfloor$ membres pour permettre une

décision, où *N* fait référence au nombre de nœuds *manager*. Il apparaît donc clairement qu'un tel processus peut créer une surcharge du trafic réseau avec un nombre élevé de *managers*, impactant alors la performance globale du *cluster*. En ce sens, nous avons opté pour un *cluster* de cinq nœuds, soit trois gestionnaires et deux travailleurs, ceci dans le but de tolérer une défaillance complète d'un nœud, quel que soit son rôle. De plus, un tel nombre de *managers* permet de respecter la contrainte d'avoir un nombre impair de nœuds gestionnaires pour que le quorum demeure toujours valide en cas de panne.

Pour permettre aux conteneurs de communiquer entre eux au sein du *cluster* Docker Swarm, un pilote réseau spécifique a été utilisé. Ce pilote permet de créer des réseaux superposés (*overlay networks*) en s'appuyant sur la technologie de tunnelisation Virtual Extensible LAN (VXLAN) (Mahalingam *et al.*, 2014) qui permet l'encapsulation de trames réseau de la couche 2 dans UDP (couche 4). Ainsi, dans un contexte de virtualisation comme celui imposé par les conteneurs, ce pilote permet de simplifier la complexité de la gestion des différents réseaux virtuels nécessaires à la communication entre les conteneurs qui sont déployés de manière éparse sur les différents nœuds du *cluster*. De plus, les réseaux superposés offrent des mécanismes qui permettent de s'assurer du bon fonctionnement de la redondance d'un point de vue réseau au sein du *cluster*. Lorsqu'un *cluster* Docker Swarm est initialisé, un réseau *ingress* est automatiquement créé sur chaque nœud. Il s'agit d'un réseau superposé particulier dont l'objectif est de faciliter la répartition de la charge des différents services entre les nœuds du *cluster*. Il est ensuite possible de créer autant de réseaux superposés que nécessaire pour interconnecter les conteneurs qui ont besoin d'échanger des données.

Par ailleurs, un réseau de type pont (*bridge network*) est également créé automatiquement sur chaque nœud lors de l'initialisation du *cluster*. Ces réseaux sont nécessaires pour relier tous les réseaux superposés (y compris le réseau *ingress*) au réseau physique des nœuds du *cluster*. Par conséquent, comme les réseaux de type pont se situent au-dessus de la couche réseau du système hôte, une communication entre les conteneurs interconnectés par un réseau superposé, mais s'exécutant dans des nœuds différents, est rendue possible. Par défaut, ces communications sont chiffrées à l'aide de l'algorithme AES-GCM (Salowey *et al.*, 2008). Les nœuds *manager* du cluster assurent le remplacement de la clé de chiffrement qu'ils partagent toutes les douze heures, ceci dans le but d'assurer un niveau de sécurité acceptable sans pour autant compromettre les performances des communications réseau.

Puisque les réseaux superposés attribuent dynamiquement une adresse IP virtuelle à chaque conteneur lors de leur création, ceux-ci sont isolés d'un point de vue du réseau, ce qui simplifie donc la consommation des services. Toutefois, lorsque le nombre de services devient important, il peut sembler pertinent de connaître à l'avance certaines VIPs, ce qui n'est évidemment pas possible. Pour résoudre ce problème, l'orchestrateur Swarm propose un mécanisme de découverte de services. Cette fonctionnalité repose sur un serveur DNS qui est intégré dans le moteur de conteneur de chaque nœud. Ce dernier est alors chargé de résoudre le nom des services pour permettre le routage des requêtes réseau vers les conteneurs correspondants grâce à leurs VIPs associées. En d'autres termes, le nom d'un service donné peut, par exemple, être utilisé en remplacement de son adresse IP virtuelle dans un fichier

de configuration. Ainsi, cette référence est interprétée correctement grâce au processus de découverte de services dès lors que la VIP est fixée.

Néanmoins, lorsque les services sont répliqués, le mécanisme de découverte de services seul n'est plus suffisant. En ce sens, grâce à l'orchestrateur, un mécanisme de répartition est mis en place dans le but de déterminer quelle réplique d'un service en particulier est utilisée. Ainsi, Docker Swarm s'appuie sur le protocole IP Virtual Servers (IPVS), une implémentation de la répartition de charge qui est déjà intégrée au niveau de la couche réseau transport du noyau Linux. Grâce à ce protocole, les requêtes TCP/UDP inter-services peuvent donc être redirigées correctement vers les conteneurs appropriés. Dans le cas spécifique d'un *cluster* Docker Swarm, chaque nœud écoute sur les ports qui sont exposés par les services qui doivent être accessibles à distance depuis un environnement extérieur (*p. ex.* un tableau de bord web qui écoute sur le port `8080`). Ensuite, IPVS applique une répartition de la charge selon l'algorithme Round-Robin (Ghaffarinejad et Syrotiuk, 2014) pour transmettre la demande à l'une des instances actives du service en question. Comme le montre la Figure 5.4, si le nœud sollicité ne contient aucune réplique du service demandé par un client, alors IPVS va router la requête à une réplique active présente sur un autre nœud, ce qui est possible grâce au réseau superposé de type *ingress*.

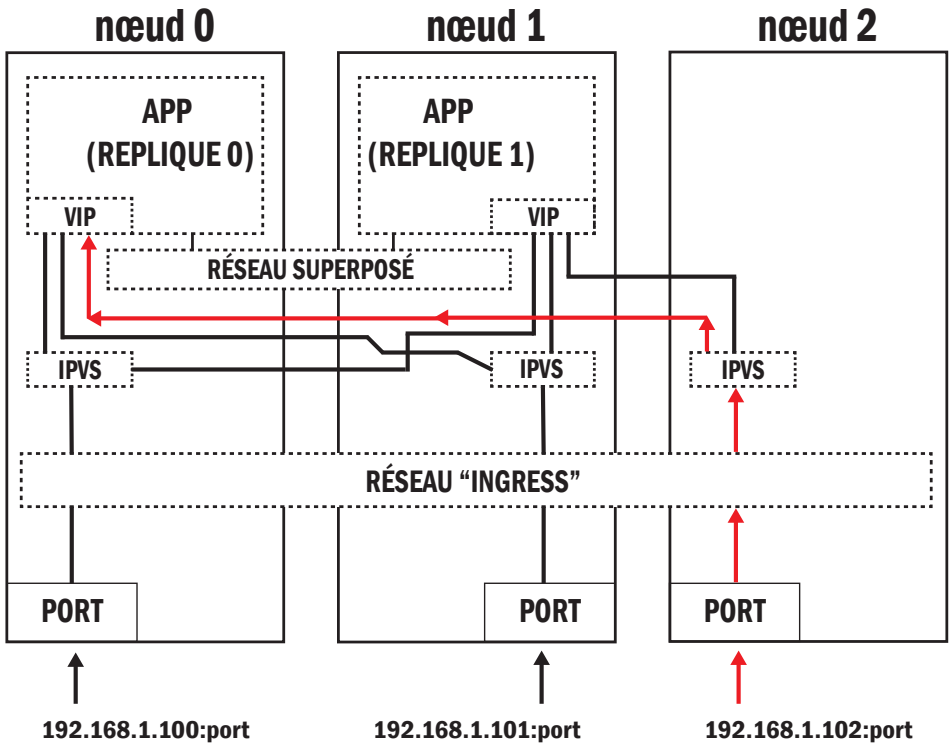


Figure 5.4 : Exemple du fonctionnement de la répartition de la charge selon l'algorithme Round-Robin appliquée par le protocole IPVS sur lequel repose Docker Swarm où un des itinéraires possibles pour une requête faite sur le service répliqué `App` est identifié par des flèches rouges.

5.1.5 PROXY INVERSE

Puisque notre architecture est composée d'un *cluster* de cinq nœuds, tous les services qui y sont déployés demeurent accessibles *via* cinq points d'entrée différents, étant donné que l'orchestrateur ouvre tous les ports de ces services sur chacun des nœuds du *cluster*. Cependant, avec un nombre important de services, il peut s'avérer difficile de gérer quel service est publié sur quel port. Ainsi, puisque la répartition de la charge au niveau des services est réalisée *via* l'orchestrateur, il semble indispensable de proposer une répartition

de la charge au niveau des nœuds également. Ceci dans le but de proposer un accès externe unifié et sécurisé aux services internes par le biais d'une URL (Uniform Resource Locator) telle que `https://service.domain.com`. Pour ce faire, l'outil open-source Træfik [25] a été préféré à d'autres options telles que NGINX [26] ou HAProxy [27], car il est le mieux adapté aux architectures basées sur des microservices. En effet, le principal avantage de Træfik est sa capacité à découvrir automatiquement les informations réseau et les services disponibles dans le *cluster* et à mettre à jour dynamiquement sa configuration en fonction de l'évolution de l'environnement. Ces fonctionnalités sont particulièrement importantes avec l'utilisation de microservices, car ceux-ci sont généralement sans état (*stateless*), mais aussi car leur durée de vie est souvent courte et de nouvelles versions peuvent avoir à être déployées fréquemment. De plus, leurs instances peuvent être mises à l'échelle de façon dynamique en fonction de l'augmentation de la demande.

Træfik est considéré comme un proxy inverse, c'est-à-dire le principal point d'entrée d'un réseau privé. En effet, les proxys inverses sont, la plupart du temps, placé derrière un pare-feu et leur rôle est d'acheminer les requêtes entrantes des clients vers un des nœuds du *cluster* et le service correspondant. En outre, les proxys inverses peuvent également remplir un certain nombre d'autres fonctions importantes pour améliorer l'efficacité et la sécurité des systèmes sur lesquels ils sont mis en place. Tout d'abord, en ce qui concerne Træfik, ce dernier

25. https://traefik.io

26. https://www.nginx.com

27. https://www.haproxy.org

fournit une répartition de la charge au niveau des nœuds ce qui permet de répartir les requêtes à l'ensemble des nœuds actifs au sein du *cluster* pour éviter que l'un d'entre eux ne soit saturé. De plus, cet outil permet également de masquer l'organisation interne de l'architecture, car il apporte une couche supplémentaire de protection en ne dévoilant jamais l'adresse IP du nœud qui traite réellement les requêtes. Enfin, toujours dans une optique d'offrir une meilleure protection, les proxys inverses, et plus particulièrement Træfik, permettent de chiffrer les échanges de données entre les clients et les services de l'architecture grâce à un ou plusieurs certificats TLS associés aux différents services ou sous-domaines. La Figure 5.5 illustre un exemple de ce fonctionnement. En effet, Træfik propose un point d'entrée unique qui permet d'accéder au service `App` *via* une URL sécurisée : `https://liara.uqac.ca/app`. Ainsi lorsque cette requête est soumise par le client (*p. ex.* un navigateur web), le répartiteur interne de Træfik achemine la requête vers l'une des répliques actives du service `App`. Dans ce scénario, la répartition de la charge fournie par l'orchestrateur Docker Swarm n'est pas représentée pour simplifier la figure, mais celle-ci demeure tout de même active et opérationnelle. En effet, bien que Træfik ait routé la requête vers le `nœud 1`, il est possible que le répartiteur de charge du *cluster*, quant à lui, achemine en interne la demande vers la `réplique 0` du service `App` qui s'exécute sur le `nœud 0`.

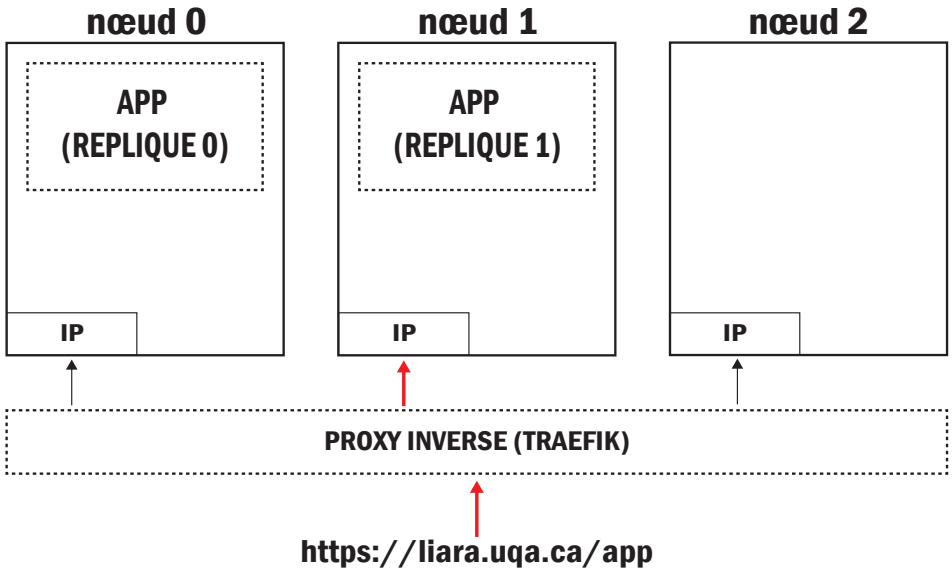


Figure 5.5 : Exemple du fonctionnement du proxy inverse Træfik lorsqu'une requête pour accéder au service `App` est effectuée par un client *via* une URL sécurisée.

5.1.6 GESTION DU *CLUSTER*

Le moteur de conteneurs Docker incluant l'orchestrateur Docker Swarm est un outil qui est utilisé à travers une interface en ligne de commandes CLI. Ainsi, il demeure possible d'initialiser et de gérer le *cluster* Docker Swarm qui régit notre architecture selon plusieurs méthodes. La plupart du temps, la méthode d'initialisation la plus appropriée consiste en l'utilisation d'un outil de gestion de configuration tel qu'Ansible [28] qui permet d'automatiser les tâches d'administration de systèmes. Cependant, comme notre architecture admet une taille raisonnable et demeure relativement simple, nous avons choisi d'initialiser le *cluster* manuellement grâce à divers scripts shell et fichiers de configuration. En effet, alors que les

28. https://www.ansible.com

outils d'automatisation offrent une meilleure gestion des flux opérationnels d'administration de systèmes complexes ainsi qu'une meilleure flexibilité globale, les scripts shell nous ont semblé être le meilleur compromis à adopter étant donné le contexte encore expérimental de cette architecture.

Par ailleurs, la gestion du *cluster* au quotidien *via* une interface en ligne de commandes a été jugée trop contraignante. De ce fait, le logiciel open-source Portainer [29] a été déployé comme un service à part entière au sein du *cluster*. Grâce à une interface graphique (Graphical User Interface ou GUI) conviviale et soignée montrée par la Figure 5.6, cet outil nous a permis

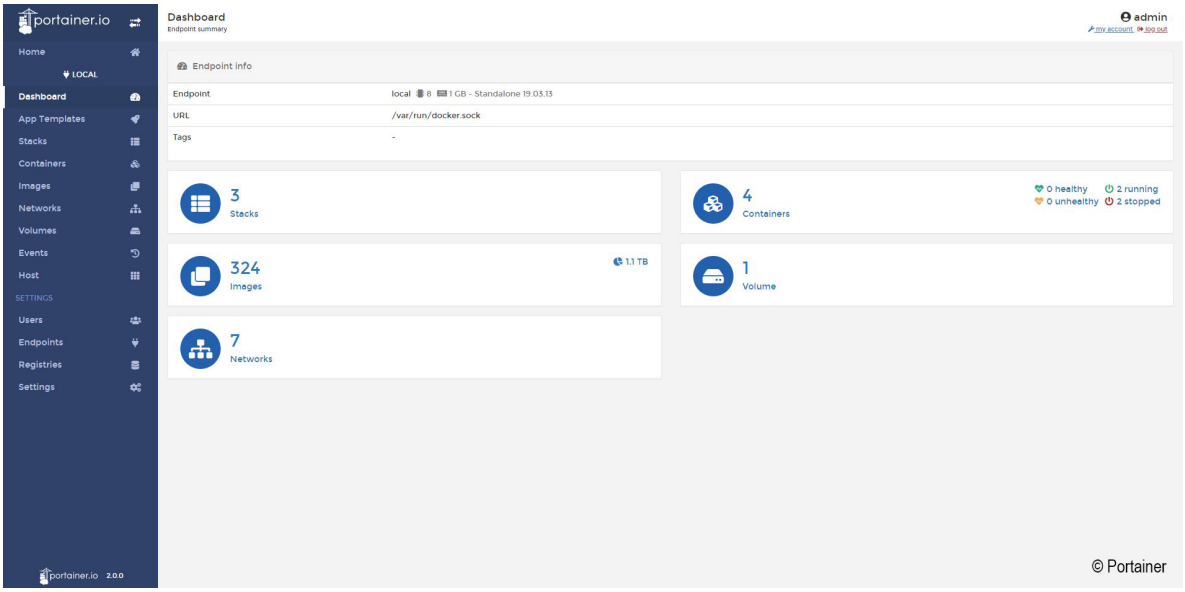


Figure 5.6 : Interface Graphique de l'outil Portainer déployé pour permettre la gestion du *cluster*.

29. https://www.portainer.io

de simplifier considérablement la tâche de gestion du *cluster*. En effet, celle-ci a permis de réduire le risque d'erreurs induit par la complexité de certaines commandes nécessaires au pilotage du moteur de conteneurs. De plus, le choix de Portainer comme outil de gestion a également été motivé par son large éventail de fonctionnalités. Les plus importantes comprennent la gestion des images, des réseaux et des volumes de conteneurs, mais cet outil propose également une visualisation des nœuds du *cluster* incluant le placement de chaque conteneur, la consultation de tous les fichiers journaux de chaque conteneur ainsi que la possibilité d'augmenter ou en diminuer le nombre de répliques pour chaque service en fonction de la charge.

5.1.7 BASE DE DONNÉES RÉPLIQUÉE

Puisque cette architecture est conçue principalement pour les maisons intelligentes, il était important de proposer un moyen fiable de stocker les valeurs des capteurs ainsi que l'état des effecteurs. Pour ce faire, trois services ont été déployés sur le *cluster*, chacun contenant une instance du Système de Gestion de Base de Données (SGBD) MongoDB [30]. Ainsi, la réplication des données a été réalisée grâce à une architecture composée d'un ensemble de répliques (*replica set*) à trois membres.

MongoDB est un SGBD open-source orienté documents et non relationnel (NoSQL), principalement utilisé pour le stockage de gros volumes de données. Un tel choix a été fait, car sa conception est axée sur la performance, la haute disponibilité et la mise à l'échelle

30. https://www.mongodb.com

automatique. Puisqu'il s'agit d'un SGBD orienté document, les données sont stockées dans des collections sous forme d'ensembles de documents. Ceux-ci font référence à une structure de données composée de paires de champs et de valeurs. En effet, ils sont formatés en Binary JavaScript Object Notation ou Binary JSON (BSON), un format de fichier similaire aux objets JSON où les champs peuvent également inclure d'autres documents, des tableaux et des ensembles de documents. Ce type de modèle de base de données dit semi-structuré implique qu'il n'existe pas de séparation entre les données et le schéma qui les représente. Ainsi, le principal avantage de l'approche orientée documents reste d'abord la possibilité de représenter des relations hiérarchiques complexes avec un seul enregistrement. Aussi, de tels modèles de données facilitent la répartition des données entre plusieurs nœuds de base de données, ce qui simplifie le processus de mise à l'échelle.

Dans le contexte de MongoDB, un ensemble de répliques constitue un groupe d'instances de base de données qui maintiennent le même ensemble de données en fournissant de la redondance ainsi qu'une haute disponibilité pour garantir la tolérance aux pannes. Le déploiement d'un tel ensemble peut se faire principalement selon deux types d'architectures différentes. Dans un premier cas, l'architecture est composée au minimum d'un nœud principal et de deux nœuds secondaires. Le second type d'architecture, quant à lui, reprend les mêmes caractéristiques du premier type, mais doit également admettre un nœud arbitre. Un nœud principal est le seul membre d'un ensemble qui tolère les opérations d'écriture. Les nœuds secondaires sont, quant à eux, en charge de préserver une copie de l'ensemble de données du nœud principal. Afin que le processus de réplication se déroule correctement, les nœuds

secondaires appliquent les instructions du journal des opérations du nœud principal à leur propre ensemble de données, et ce, de manière asynchrone. En revanche, dans une architecture d'ensemble de répliques qui contient un arbitre, ce dernier ne dispose pas de copie de l'ensemble de données du nœud principal. Ainsi, le nœud arbitre ne peut, d'aucune façon, devenir un nœud principal en cas de panne de ce dernier. Toutefois, le nœud arbitre participe à l'élection d'un nouveau nœud principal lorsque cela est nécessaire. En ce sens, dans le cas d'un ensemble de répliques à trois membres, la mise en œuvre d'une architecture composée d'un nœud principal et de deux nœuds secondaires a été préférée. En effet, le fait de remplacer un nœud secondaire par un arbitre ne permet pas de garantir une aussi bonne répartition de la charge lors des opérations de lecture des données.

La Figure 5.7 illustre le fonctionnement de cette architecture d'ensemble de répliques composée d'un nœud principal et de deux nœuds secondaires (P-S-S). Toutes les deux secondes, tous les nœuds s'envoient des pouls (`A`) ; dans le cas où l'un d'entre eux ne revient pas à son émetteur dans les dix secondes, les nœuds restants identifient alors le nœud non réactif comme inaccessible (`B`). Lorsque c'est le nœud principal qui est indisponible, un processus d'élection est alors déclenché pour que l'un des deux nœuds secondaires devienne le nouveau nœud principal (`C`). De la même manière que pour l'orchestration du *cluster*, la recommandation qui est formulée dans la documentation pour déployer un ensemble de répliques MongoDB est de mettre en place un nombre total de nœuds impair. Ainsi, il est possible, dans le cas d'une architecture P-S-S comme celle-ci, de subir la perte totale d'une

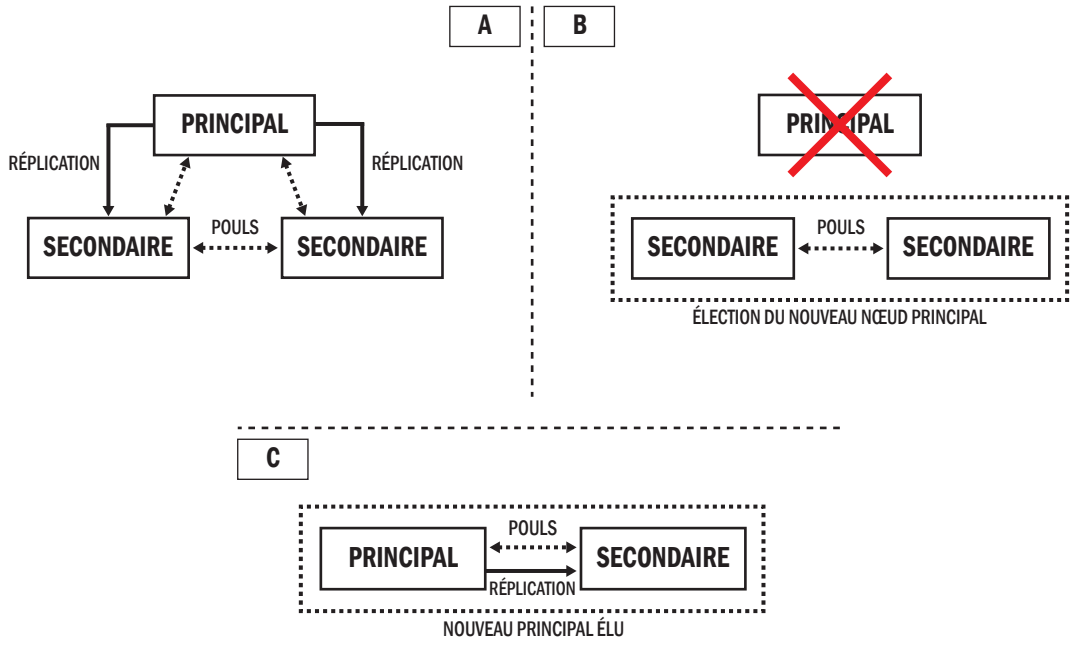


Figure 5.7 : Processus d'élection avec la base de données MongoDB selon l'architecture d'ensemble de répliques à trois nœuds de type P-S-S où `A`, `B` et `C` représentent respectivement les étapes successives lors de la perte totale du nœud principal.

instance de la base de données. En effet, les quorums de tolérance aux pannes demeurent identiques pour les deux composants logiciels.

5.1.8 DÉPÔT D'IMAGES DE CONTENEURS

Le principal avantage de l'utilisation d'une architecture de microservices, en particulier dans un environnement comme celui des laboratoires de recherche dans le domaine des habitats intelligents, est de fournir un moyen simple de déployer des composants logiciels expérimentaux qui fonctionnent dans des conteneurs pour réaliser la reconnaissance d'activités. Ainsi, dans le but de faciliter la gestion de versions (*versionning*), la distribution et la

réutilisation de ces composants, un système de stockage des images des conteneurs a été mise en place au sein du *cluster*.

Les images sont des fichiers immuables qui sont composés de plusieurs couches contenant le code source, les librairies, les dépendances ainsi que les outils nécessaires au bon fonctionnement des conteneurs. En d'autres termes, les images décrivent les services et leur environnements virtuels à un instant donné. En fin de compte, les conteneurs ne représentent que des images qui sont en cours d'exécution. En ce sens, celles-ci peuvent être stockées dans un dépôt, où les répertoires qu'il admet contiennent les différentes versions pour chacune des images. Grâce à un tel registre, il devient possible pour les expérimentateurs de télécharger certaines images (*pull*), d'en créer (*push*) de nouvelles contenant leurs propre composants logiciels ou de créer de nouvelles versions d'images déjà existantes. Par défaut, le moteur de conteneurs installé sur chaque nœud de l'architecture est configuré pour utiliser le dépôt public DockerHub [31]. Cependant, puisque les applications expérimentales conçues dans les environnements de recherche peuvent contenir des vulnérabilités, des informations confidentielles ou un certain nombre d'innovations, la configuration du moteur de conteneurs a été modifiée afin que ce dépôt privé d'images puisse être utilisé dans le *cluster*.

5.1.9 ORGANISATION DES CONTENEURS

Pour résumer les différents aspects qui décrivent la conception de l'architecture proposée et qui ont été introduits dans ce chapitre, la Figure 5.8 propose une représentation graphique

31. https://hub.docker.com

du déploiement des conteneurs sur les nœuds qui composent le cluster. De plus, les réseaux superposés pour chaque service y sont également illustrés.

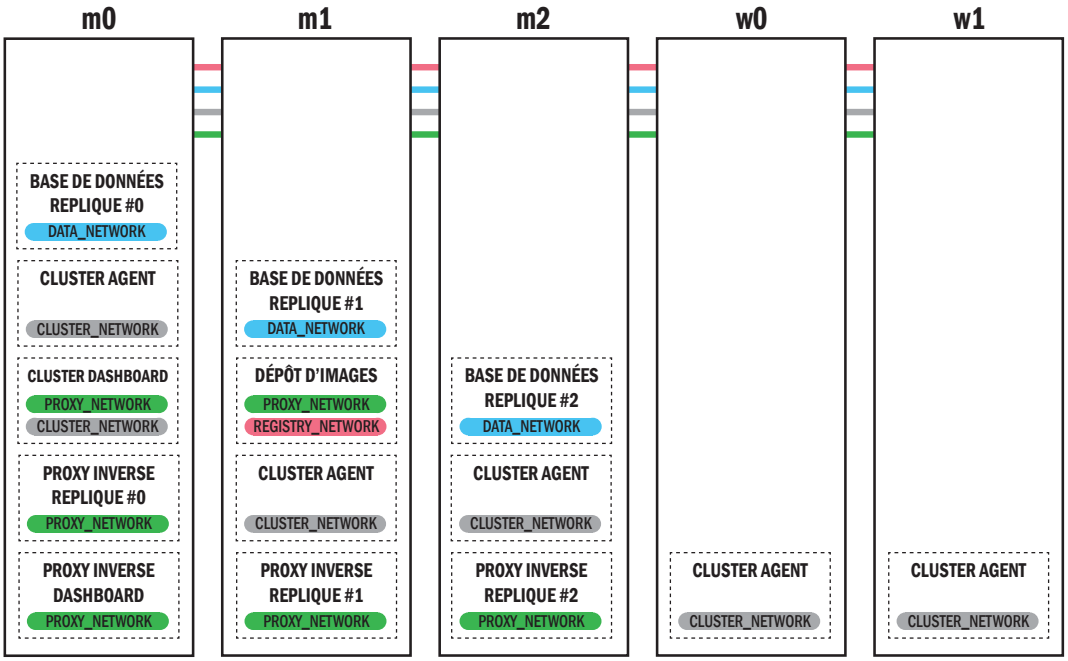


Figure 5.8 : Déploiement des conteneurs sur les nœuds du *cluster* qui décrivent la conception de l'architecture proposée.

Dans ce déploiement, il est possible de voir que l'application Træfik admet deux services distincts. Le premier concerne le service de proxy inverse dont trois répliques sont exécutées sur chacun des nœuds de gestion du *cluster* (`m0`, `m1`, `m2`). Ce placement a été privilégié, car cela permet de s'assurer de la fiabilité d'un élément aussi important en cas de perte d'un *manager*. Le second service est, quant à lui, le tableau de bord web qui permet de visualiser, en temps réel, les routes actives vers les différents services ouverts à un environnement extérieur qui sont détectées par l'outil. De plus ce tableau de bord permet de visualiser plusieurs statistiques

d'accès pertinentes pour l'évaluation de la charge en fonction des services. Néanmoins, puisque ce tableau de bord n'a pas un impact majeur sur le bon fonctionnement du cluster, son déploiement n'admet aucune réplique supplémentaire au seul conteneur qui lui est attribué. Toutefois, ce service est doté d'une politique de redémarrage qui prévoit de toujours relancer le conteneur si son exécution est interrompue. En ce sens, s'il se produit une défaillance au niveau du nœud sur lequel il est déployé (`m0`), l'exécution du conteneur du tableau de bord de Træfik peut être relancée soit sur `m1`, soit sur `m2`. De la même manière, des instructions de déploiement identiques ont été définies pour le service du dépôt d'images et le service du tableau de bord de l'outil de gestion du *cluster*.

Par ailleurs, en ce qui concerne la base de données, l'ensemble de répliques MongoDB a été déployé sur les nœuds de gestion dans le but de fournir, en permanence, une réplication sûre des données et de tolérer, au maximum, la défaillance d'un unique nœud. Enfin, puisque le rôle de l'outil Portainer est de pouvoir gérer l'intégralité du *cluster*, un agent, c'est-à-dire, un service global a été déployé sur chaque nœud pour permettre à l'application web de transmettre les bonnes requêtes aux différents moteurs de conteneurs présents sur chacun des nœuds.

En ce qui concerne la communication réseau, un réseau superposé a été créé pour chaque service. De plus, le tableau de bord du proxy inverse est automatiquement exposé et accessible *via* l'URL suivante : `https://liara.uqac.ca/proxy`. Toutefois, puisque le tableau de bord de l'outil de gestion du *cluster* et le dépôt d'images doivent également être accessibles depuis l'extérieur du *cluster*, ces services sont connectés au réseau superposé du proxy inverse. Ainsi, le routage des demandes clients en ce qui concerne les accès à ces

services est rendu possible *via* les URLs suivantes : `https://liara.uqac.ca/cluster` et `https://liara.uqac.ca/registry`.

5.2 EXPERIMENTATIONS

Afin d'évaluer et de valider l'architecture introduite dans les sections précédentes, une installation expérimentale préliminaire a été déployée dans notre laboratoire, le LIARA. En outre, plusieurs expérimentations ont été réalisées afin de confirmer la capacité pour ce système à résister à différentes défaillances qui pourraient survenir afin d'en déterminer sa fiabilité globale. Pour ce faire, cette section commence par décrire la mise en œuvre matérielle utilisée pour le fonctionnement de l'architecture. Une seconde section décrit les expérimentations qui ont été menées ainsi que les résultats qui en découlent.

5.2.1 INSTALLATION MATÉRIELLE

Afin de minimiser les difficultés dans la reproduction du déploiement de l'architecture présentée de ce chapitre ainsi que les expérimentations qui y ont été appliquées, la mise en œuvre de cette architecture a été faite en utilisant du matériel open-source. Pour ce faire, notre choix s'est basé sur l'utilisation de sept SoCs Raspberry Pi 3 Model B, un routeur sans fil conventionnel (LinkSys WRT1900AC) fonctionnant avec le système d'exploitation OpenWRT [32] ainsi qu'un simple commutateur réseau. Le principal avantage de l'utilisation des Raspberry Pi est qu'elles offrent un rapport qualité-prix intéressant pour les besoins de

32. https://openwrt.org

cette architecture. En effet, alors que leur coût unitaire est de 35 dollars US, celles-ci intègrent un processeur ARMv8 quadricœur de 1,2 GHz et 1 GiB de SDRAM. De plus, pour que ce déploiement expérimental reste aussi peu onéreux que possible, le stockage n'a pas été effectué sur des systèmes RAID. Au lieu de cela, deux clés USB ont été connectées aux deux Raspberry Pi utilisées pour le système de fichiers distribué (`f0` et `f1`).

De plus, comme ce modèle en particulier de Raspberry Pi dispose d'un processeur basé sur l'architecture ARMv8, l'utilisation de systèmes d'exploitation 64-bits y est supportée. En ce sens, nous avons opté pour une version spécifique de la distribution Linux Debian [33] qui a été modifiée selon nos besoins [34]. Ce choix a été établi considérant que le système d'exploitation Raspberry Pi OS [35] n'est toujours pas disponible en version 64-bits malgré la sortie de nouvelles Raspberry Pi disposant de plusieurs gibibits de SDRAM. Puisque les conteneurs virtualisent leurs environnements au niveau du système hôte, l'emploi d'un système d'exploitation 64-bits était péremptoire. En effet, plusieurs outils qui sont requis dans la conception de notre architecture étaient disponibles uniquement pour l'architecture de processeur ARMv8 (*p. ex.* MongoDB).

---

33. https://www.debian.org

34. https://github.com/FlorentinTh/FlOS

35. https://www.raspberrypi.org/downloads/raspberry-pi-os/

5.2.2 HAUTE DISPONIBILITÉ

Suivant l'initialisation du *cluster* Docker Swarm sur chacun des cinq nœuds de l'architecture (`m0`, `m1`, `m2`, `w0`, `w1`), tous les composants applicatifs y ont été déployés selon la politique de placement définie en Section 5.1.9. Ainsi, l'architecture distribuée a été opérationnelle en seulement quelques minutes. Cette politique de déploiement a été spécifiquement déterminée pour supporter la défaillance totale d'un nœud au maximum. Cependant, bien que l'ajout de nœuds supplémentaires au *cluster* doit irréfutablement améliorer la tolérance aux pannes, avoir tous ces nœuds dans le même emplacement géographique n'améliorent, en réalité, pas la fiabilité de l'ensemble du système. De plus, cela créera un environnement beaucoup plus complexe à gérer et à maintenir. Par conséquent, les différentes expérimentations réalisées sur cette architecture ont pour but de confirmer que cette implémentation demeure fiable.

Le premier protocole expérimental mis en place est relativement simple. La défaillance d'un nœud a été simulée en débranchant son câble Ethernet au niveau du commutateur réseau afin que celui-ci ne puisse plus communiquer avec le reste du *cluster*. En ce sens, nous avons commencé par déconnecter un seul nœud de type *worker* (`w0`). Cette expérimentation est identifié : *F0*. Comme qu'attendu, tous les services déployés au sein de l'architecture sont restés accessibles et tous les conteneurs qui fonctionnaient sur `w0` ont été automatiquement répartis et relancés sur les nœuds opérationnels restants. Enfin, après avoir reconnecté ce nœud au réseau, il a montré un état opérationnel en quelques secondes. Cependant, les conteneurs qui s'exécutaient auparavant sur `w0` n'ont pas été redémarrés sur ce nœud d'origine, car ceux-ci étaient pleinement opérationnels sur les autres nœuds. Dans un second temps, lorsqu'un

unique nœud de gestion (*manager*) a été débranché du réseau (`m2`), les mêmes résultats ont été observés (*F1*). En revanche, lorsque le nœud leader (`m0`) a été privé de sa connexion réseau, le processus d'élection s'est enclenché et c'est le nœud `m2` qui a été choisi comme nouveau leader (*F2*). Ensuite, deux nœuds de différents types ont été débranchés du commutateur réseau, soit un nœud *manager* (`m1`) ainsi qu'un *worker* (`w0`), tandis que les autres sont restés connectés (*F6*). Bien qu'avec cette architecture notre objectif était de tolérer la défaillance d'un nœud uniquement, le bon fonctionnement du *cluster* n'a pas été altéré dans cette situation. Cela peut s'expliquer par le fait que le dysfonctionnement a été simulé sur deux nœuds ayant un rôle différent. En effet, la reproduction de cette expérimentation en débranchant le câble réseau de deux nœuds gestionnaires incluant le leader (*p. ex.* `m0` et `m1` ou `m0` et `m2`) a permis de constater que l'intégrité de l'ensemble de l'architecture était compromise (*F7*). Bien que les services démarrés sur les nœuds travailleurs ont continué leur exécution normalement, la capacité, au sein du *cluster*, d'effectuer des tâches de gestion, y compris le démarrage de nouveaux services était perdue. En ce sens, cette expérimentation nous permet d'affirmer qu'étant donné la mise en œuvre proposée, il demeure plus sûr de garder à l'esprit que la défaillance d'un seul nœud reste la limite maximale à supporter, puisque selon leur type, la perte de deux nœuds peut potentiellement compromettre le bon fonctionnement du *cluster*.

Dans un second temps, la procédure expérimentale précédente a été reproduite en simulant la défaillance des nœuds *via* une déconnexion de leur alimentation électrique pour les arrêter, au lieu de les déconnecter du réseau (*F3*, *F4*, *F5*). Bien que des résultats similaires aient été observés, le temps de récupération qui a été constaté pour chaque nœud était plus important,

c'est-à-dire, quelques minutes au lieu de quelques secondes. De plus, une expérimentation supplémentaire a été menée afin d'évaluer la résilience du *cluster* en cas de coupure générale de l'alimentation électrique (*F8*). Pour ce faire, toutes les Raspberry Pi ont été mises hors tension à l'aide du disjoncteur présent sur la multiprise à laquelle elles étaient branchées. Le routeur et le commutateur réseau, quant à eux, sont restés sous tension. Ainsi, puisque l'état du *cluster* est préservé grâce à un mécanisme interne du moteur de conteneurs, le *cluster* a pu restaurer son état de fonctionnement initial lui-même, après le redémarrage de chaque nœud et le rétablissement de la communication inter-gestionnaires. Le temps de total récupération observé pour une telle coupure générale a été d'environ une dizaine de minutes. Le Tableau 5.1 présente un récapitulatif de ces expérimentations ainsi que les résultats qui ont été obtenus.

Tableau 5.1 : Récapitulatif des expérimentations réalisées sur l'architecture afin de déterminer sa fiabilité globale.

	F0, *F3*	*F1*, *F4*	*F2*, *F5*	*F6*	*F7*	*F8*
Nœud(s)	`w0`	`m2`	`m0`	`m1` & `w0`	`m0` & `m1`	`tous`
Type de panne	réseau - électrique	réseau - électrique	réseau - électrique	réseau - électrique	réseau - électrique	électrique
Tâches relancées	oui	oui	oui	oui	non	non
Élection d'un leader	non	non	oui	non	non	non
État du cluster	disponible	disponible	disponible	disponible	H.S.	disponible
Temps de récupération	$\approx 1\,s$ - $\approx 1\,m$	$\approx 1\,s$ - $\approx 1\,m$	$\approx 1\,s$ - $\approx 1\,m$	$\approx 1\,s$ - $\approx 1\,m$	-	$\approx 10\,m$

Finalement, ces deux expériences ont prouvé que l'architecture proposée dans ce chapitre offre une fiabilité globale plus que satisfaisante. En effet, selon les situations de défaillances, elle a su démontrer sa capacité à toujours supporter la défaillance d'un nœud. Cependant, comme les pannes électriques peuvent impliquer un temps de récupération plus long, nous suggérons que tout l'équipement nécessaire au déploiement de cette architecture soit alimenté électriquement par une source d'Alimentation Sans Interruption (ASI).

5.3 CONCLUSION

Dans ce chapitre, une nouvelle architecture pour les maisons intelligentes a été proposée. Par opposition aux architectures existantes, cette implémentation vise à être agnostique vis-à-vis de la conception des applications qu'elle admet, mais également fiable et évolutive. Ainsi, la conception qui a été proposée repose sur le concept de microservices mis en œuvre grâce à un *cluster* composé de cinq nœuds distribués. L'utilisation de microservices permet d'améliorer à la fois l'interopérabilité entre les applications d'apprentissage machine qui constituent généralement un ensemble hétérogène de technologies, ainsi que leur réutilisabilité.

De plus, cette implémentation a été conçue pour être compatible avec la majorité des architectures de maisons intelligentes existantes, car elle peut facilement remplacer l'unité centrale de calcul (serveur ou un *broker*) que celles-ci admettent. Aussi, cette architecture s'inscrit dans la continuité de l'objectif qui consiste à supprimer complètement tout point de défaillance unique au sein des habitats intelligents afin que ceux-ci puissent offrir un niveau de fiabilité plus élevé à leurs résidents.

Enfin, le dernier objectif de cette implémentation concerne la possibilité d'y ajouter des ressources supplémentaires, tant matérielles que logicielles, en fonction des besoins. Ainsi, cette architecture permet, lorsqu'elle est déployée dans un environnement de recherche, un plus grand niveau d'évolutivité, ce qui permet alors de simplifier les procédures d'expérimentations. Grâce aux évaluations qui ont été réalisées pour évaluer la haute disponibilité de cette architecture, nous avons montré que le bon fonctionnement de l'ensemble du système n'était pas affecté lors d'un dysfonctionnement complet d'un nœud du *cluster*. De plus, ces expérimentations ont également permis de révéler que l'architecture était également résistante à une panne de courant généralisée. Néanmoins, puisque le temps de récupération du *cluster* après une panne électrique demeure important, il est important de suggérer l'utilisation d'une alimentation sans interruption pour prévenir ce genre de situations.

CHAPITRE VI

LE2ML : UN *WORKBENCH* MODULAIRE POUR L'APPRENTISSAGE MACHINE

Dans le chapitre précédent, une nouvelle architecture d'habitat intelligent a été présentée. Dans sa conception, cette implémentation suggère de s'appuyer sur la technologie des microservices puisque ceux-ci offrent un meilleur niveau de flexibilité et de modularité. Ainsi, cette technologie permet donc d'abstraire le type de capteurs (ambiants ou *wearable devices*) qui est utilisé pour le processus de reconnaissance d'activités. Cependant, pour réaliser la reconnaissance d'activités avec des *wearable devices*, la plupart des méthodes actuellement proposées sont des applications complètes. Par conséquent, les différents processus qui composent la reconnaissance sont, la plupart du temps, encapsulés au sein d'un unique composant logiciel immuable. Il devient donc difficile de les modifier. De plus, le fait de disposer d'une unique application ne favorise pas la réutilisation de certains mécanismes communs à plusieurs méthodes. Ceux-ci doivent donc généralement être soit développés à nouveau, soit adaptés afin de pouvoir supporter des modifications dans la méthode proposée. De surcroît, l'exploitation de ce genre d'application « prête à l'emploi » est, la plupart du temps, complexifiée. En effet, selon les dépendances, le langage de programmation et la plateforme, elles peuvent être difficiles à adapter et à déployer d'un environnement à un autre (*p. ex.* entre deux laboratoires de recherche différents). De ce fait, la question de recherche qui est spécifiquement examinée dans ce chapitre est : *« comment prendre en considération la diversité des composants logiciels, et plus précisément, ceux exploités par les wearable devices, qui composent les*

différents processus d'apprentissage en facilitant leur intégration, leur réutilisation ainsi que leur déploiement au sein de l'architecture ? »

Pour répondre à cette question, ce chapitre présente LE2ML (LIARA Environment for Modular Machine Learning), un nouveau type de *workbench* pour l'apprentissage machine qui repose sur l'utilisation de microservices établis par l'architecture introduite au chapitre précédent. De la même manière, cette conception logicielle a été choisie, car la technologie des microservices permet une meilleure évolutivité, des déploiements plus sûrs et plus rapides ainsi qu'une meilleure isolation des pannes (Dragoni *et al.*, 2017). Cependant, l'avantage principal de cette solution concerne l'aspect modulaire que permettent les microservices. De plus, le souhait de proposer une telle conception a été amplifié par la volonté de finaliser l'intégration logicielle des *wearable devices* au sein de l'architecture en *cluster* présentée précédemment. Cependant, puisque ce *workbench* demeure un outil agnostique, tant en termes de langages de programmation que de plateformes supportées, qui peut être déployé dans n'importe quel environnement, et ce, sans aucune contrainte complexe ; les habitats intelligents qui n'exploitent pas les *wearable devices* pourraient malgré tout tirer profit d'une telle solution.

La suite de ce chapitre commence par présenter le fonctionnement du *workbench* LE2ML. Ensuite, une expérimentation pour évaluer la reproductibilité de l'expérimentation proposée dans le Chapitre 4 et valider les résultats attendus est décrite dans une troisième section. Finalement, dans une dernière partie, ce chapitre dresse une conclusion quant à ce dernier travail.

6.1 SOLUTION PROPOSÉE

Dans le Chapitre 2, les trois principaux *workbench* d'apprentissage machine ont été présentés dans le but de déterminer les problématiques de chacune de ces solutions. Ainsi, il a été montré que tous ces outils présentent le même principal inconvénient, c'est-à-dire, une forte dépendance à leur contexte d'utilisation. En effet, bien que toutes ces solutions permettent d'intégrer des composants logiciels personnalisés, ceux-ci doivent être conçus pour répondre aux exigences de l'outil pour lequel ils sont développés. De plus, dans un contexte de recherche académique, ces paquets additionnels peuvent devenir difficiles à maintenir. En effet, ils doivent être installés manuellement sur chacun des dispositifs qui composent l'environnement (*p. ex.* ordinateurs ou serveurs), ce qui provoque indubitablement des problèmes de déploiement et de réutilisation.

Par conséquent, ce chapitre présente LE2ML [36] (LIARA Environment for Modular Machine Learning), un nouveau type de *workbench* d'apprentissage machine qui, comme pour l'architecture introduite précédemment, s'appuie sur la technologie des microservices. Bien que cet outil ait été spécifiquement conçu pour être utilisé dans notre architecture, il est parfaitement possible de le faire fonctionner dans des environnements plus traditionnels tels qu'un serveur dans une architecture de maison intelligente monolithique ou un simple ordinateur. La seule contrainte préalable est d'y installer le moteur de conteneurs Docker. LE2ML a été développé dans le but d'être modulaire. De ce fait, le composant principal de la

36. https://github.com/FlorentinTh/LE2ML

conception de son architecture logicielle est une API (Application Programming Interface) REST. Cette API est principalement chargée de piloter tous les autres composants logiciels du *workbench* qui sont considérés comme des modules que n'importe qui peut développer afin d'intégrer à l'outil de nouvelles fonctionnalités selon les besoins. Enfin, une application web a également été développée pour faciliter les interactions avec l'API. La Figure 6.1 présente un exemple de déploiement possible pour LE2ML. En effet, celle-ci illustre l'organisation des conteneurs et de leurs réseaux superposés qui sont nécessaires pour que le *workbench* fonctionne sur notre architecture. Ainsi, cette figure va guider les explications fournies dans le reste de ce chapitre.

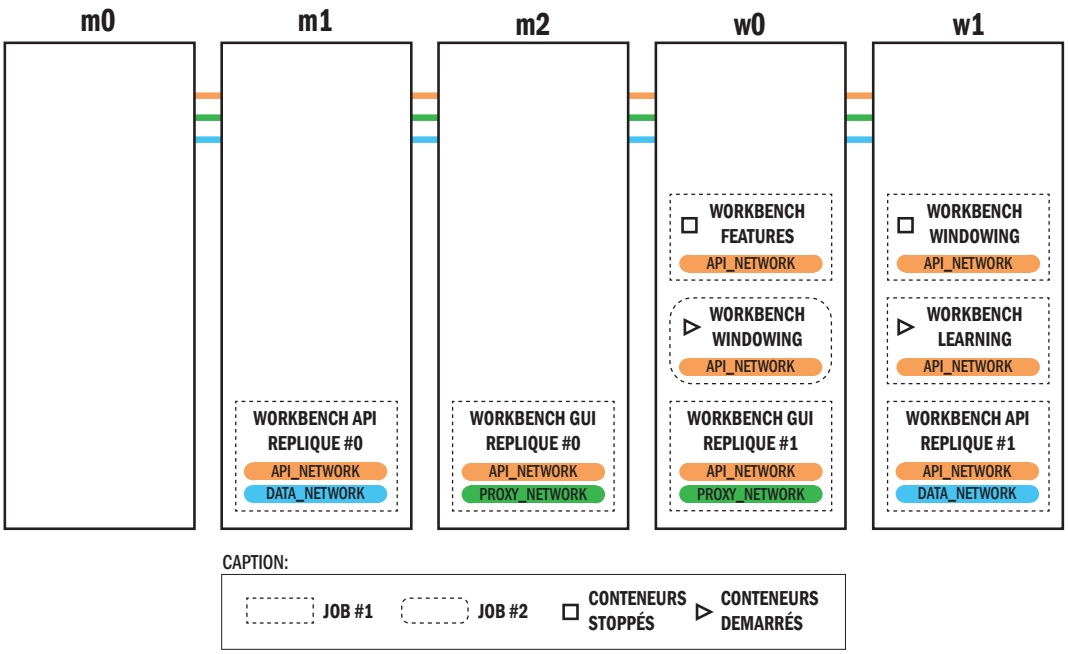


Figure 6.1 : Exemple de placement des conteneurs et leurs réseaux superposés sur chaque nœud de l'architecture proposée précédemment lors du déploiement de LE2ML.

6.1.1 API REST

En tant que composant logiciel principal dans la conception de LE2ML, l'API [37], a été développée grâce à Node.js [38]. Cet outil constitue un environnement d'exécution basée sur le moteur V8 [39] qui permet de créer des applications côté serveur en JavaScript. Ensuite, le *framework* Express.js [40] a également été employé pour la création de l'API REST. Plus précisément, celui-ci a permis de simplifier la définition des fonctions de traitement associées aux requêtes HTTP qui répondent aux différentes URI de l'application (cf. Section 3.4.3).

Par ailleurs, comme le montre la Figure 6.1, cette API a été déployée en tant que service utilisant deux répliques. Celles-ci peuvent être exécutées soit sur des nœuds de type *manager* ou sur des *workers*. Cet exemple de déploiement demeure la recommandation suggérée puisque l'accessibilité de l'API est garantie, même en cas de défaillance d'un nœud de l'architecture. Toutefois, en fonction de la demande en ressource, c'est-à-dire, lorsque l'API doit traiter un grand nombre de requêtes qui sont produites par plusieurs utilisateurs, il est possible de procéder à une mise à l'échelle du service afin d'éviter un ralentissement de l'ensemble du *workbench*. De plus, bien que le service de l'API fournisse son propre réseau superposé (`api_network`), l'ensemble de ses répliques doivent aussi être relié au réseau de données (`data_network`) afin de pouvoir communiquer avec la base de données déployée au sein de

37. https://github.com/FlorentinTh/LE2ML-API

38. https://nodejs.org

39. https://v8.dev

40. https://expressjs.com

l'architecture. En effet, l'API doit principalement être en mesure de manipuler les données relatives aux utilisateurs, tâches et sources de données, mais également aux algorithmes d'apprentissage machine, de fenêtrage et d'extraction de caractéristiques qui sont utilisés dans les modules de LE2ML. Bien que cette implémentation a été préférée dans le cadre d'une utilisation avec l'architecture présentée dans le chapitre précédent, la mise à disposition d'une base de données non relationnelle n'est cependant pas une contrainte imposée pour assurer le fonctionnement du *workbench* dans des environnements différents.

L'objectif principal de l'API de LE2ML est d'exécuter plusieurs processus (*jobs*) d'apprentissage machine simultanés grâce à la définition de *pipelines* pour ce processus. Pour ce faire, l'API est chargée d'orchestrer le démarrage des différents modules du *workbench* qui s'exécutent alors à l'intérieur de conteneurs. Lorsque tous les modules ont terminé, le *pipeline* d'apprentissage machine est complété et la performance de la reconnaissance est donnée selon plusieurs métriques d'évaluation. De plus, l'API propose également plusieurs fonctionnalités de base qui ne sont pas des modules, car elles sont complémentaires au processus d'apprentissage machine. Celles-ci comprennent l'importation de fichiers de données, la gestion de ces fichiers importés ou nouvellement créés, la visualisation des données, la gestion de l'authentification et des autorisations pour les utilisateurs et les applications, la gestion des utilisateurs et de leur rôle ainsi que la gestion des modules.

La définition d'un *pipeline* d'apprentissage machine se fait par le biais d'un fichier Yet Another Markup Language (YAML). Cette approche a été choisie, car un tel type de fichier demeure plus facile à lire et à écrire que le format JSON. Par ailleurs, le YAML est un format

qui se prête particulièrement bien avec l'utilisation de systèmes de gestion de versions (Version Control Systems ou VCSs) pour être partagés entre plusieurs expérimentateurs au sein d'une équipe de recherche. Une fois transmis à l'API, le fichier de définition du *pipeline* est converti en format JSON afin d'être validé grâce à un schéma, puis il est stocké sur le système de fichiers distribué. Ensuite, le contenu du fichier est analysé par l'API afin que toutes les tâches qui composent le *pipeline* soient lancées de manière consécutive. Un exemple d'un tel fichier de définition est présenté en Figure 6.2.

```yaml
version: '1'
pipeline: machine_learning
source: inertial
process: train
model: model-202009222020
cross-validation: true
input:
  file:
    type: raw
    filename: soil_9_axis_imu_lsm9ds1.csv
windowing:
  enable: true
  parameters:
    length: 360Hz
    function:
      label: rectangular
      container: core-windowing
    overlap: 0
features:
  save: true
  filename: saved-features.csv
  list:
    ...
    - label: kurtosis_adjusted
      container: core-inertial-features
    ...
algorithm:
  name: k_nearest_neighbors
  container: core-py-sk
  parameters:
    search_algorithm: linear
    num_neighbors: 1
    distance: manhattan
```

Figure 6.2 : Exemple d'un fichier de définition d'un *pipeline* qui décrit la phase d'entraînement d'un processus d'apprentissage machine traditionnel.

Dans un premier temps, l'utilisateur doit préciser le type de *pipeline* qui doit être réalisé. Les possibilités sont soit apprentissage machine (`machine_learning`), soit apprentissage profond (`deep_learning`). Une autre propriété obligatoire concerne la source de données. Celle-ci correspond au type de données qui sont traitées au cours du processus (*p. ex.* des données inertielles ou vocales). Par ailleurs, lorsqu'il s'agit d'un *pipeline* définissant un processus d'apprentissage machine traditionnel, la propriété `process` doit également être

déclarée. Les valeurs possibles sont `train` et `test` qui font respectivement référence aux phases d'entraînement et de test pour un tel processus. Il est également possible d'y attribuer la valeur `none`—ceci dans le but de permettre aux utilisateurs de réaliser uniquement certaines tâches qui précèdent le processus d'apprentissage, comme l'extraction de caractéristiques. Comme le montre cet exemple, dans le cas où le *pipeline* doit accomplir une phase d'entraînement, l'attribut `model` doit obligatoirement être fourni. Ainsi, le modèle de données qui a été entrainé par l'algorithme d'apprentissage est sauvegardé sur le système de fichier pour être réutilisé ultérieurement (*p. ex.* lors de la définition d'une phase de prédiction). De plus, la propriété `cross-validation` est la dernière propriété imposée lorsqu'il s'agit d'une phase d'entraînement. Elle permet de produire une estimation des performances de reconnaissance en évaluant le modèle nouvellement créé grâce à la méthode de la validation croisée en 10 plis.

Ensuite, il est nécessaire de déclarer les données d'entrée sur lesquelles est appliqué le processus d'apprentissage machine. Actuellement, seules deux options sont autorisées, à savoir, la définition d'un fichier ou une connexion WebSocket. Lorsqu'il s'agit d'une source d'entrée de type WebSocket, l'URL du serveur doit être indiquée. Sinon, les fichiers doivent être référencés selon leur nom et le type de données qu'ils contiennent (*p. ex.* données brutes : `raw`). Enfin, le reste du contenu du fichier permet de décrire les différentes tâches et leurs paramètres associés qui sont utilisés lors de l'exécution des modules.

Pour compléter l'exécution des *jobs*, l'API a la responsabilité de démarrer les conteneurs selon la définition des tâches fournie dans les fichiers de définition de *pipelines*. Ainsi, puisque certains conteneurs dépendent du résultat de la tâche précédente, ils sont, pour un *job* donné,

lancés les uns après les autres. Néanmoins, lorsqu'une même tâche est définie pour utiliser plusieurs conteneurs distincts, ceux-ci sont démarrés en même temps. Ils s'exécutent alors en parallèle. Enfin, les *jobs* qui sont créés par les différents utilisateurs du *workbench* sont également exécutés en parallèle. Ceux-ci sont aussi indépendants les uns des autres bien qu'ils utilisent des images de conteneurs identiques. À titre d'exemple, la Figure 6.1 montre deux *jobs* démarrés à des moments différents, mais dont le traitement est concurrent. En effet, il est possible d'observer que pour le premier *job*, les tâches de fenêtrage et d'extraction de caractéristiques sont toutes les deux terminées alors que le conteneur d'apprentissage est toujours en cours d'exécution. À l'inverse, la tâche de fenêtrage pour le second *job* vient juste de commencer et les autres conteneurs ne sont pas visibles, car ils n'ont pas encore été démarrés.

Pour qu'un conteneur devienne un module de LE2ML, trois variables d'environnement doivent être définies, lors de son lancement. Il s'agit de l'identifiant unique du *travail*, de l'identifiant unique de l'utilisateur ainsi que d'un jeton unique généré aléatoirement pour chaque conteneur. Le jeton est utilisé pour permettre à l'API de faire le lien entre un conteneur en particulier et la tâche à laquelle il est rattaché. Par conséquent, il est possible de mettre à jour l'entrée du *job* associé dans la base de données. En effet, une fois son traitement terminé, un module LE2ML doit communiquer son état (succès ou échec) à l'API par le biais d'un *endpoint* prévu à cet effet. De telles requêtes ne sont permises que si les conteneurs y sont autorisés. Cette vérification est établie grâce à une clé d'application générée au préalable pour chaque application.

Finalement, comme l'exécution des processus d'apprentissage machine implique la création de fichiers, ces informations résultantes du traitement de chaque tâche du *pipeline* sont écrites sur le système dans des dossiers séparés par *job*. En ce sens, ces dossiers sont montés en tant qu'espaces de travail pour tous les conteneurs définis pour un *job* donné.

6.1.2 APPLICATION WEB

Pour faciliter l'interaction avec l'API, nous avons jugé approprié de proposer une interface. De ce fait, une interface graphique [41] a été préférée à une interface de type CLI (dont le développement n'est pas exclu dans un avenir proche). En effet, puisque les équipes de recherche sont, la plupart du temps, multidisciplinaires, la conception d'une GUI a été priorisée, car ce type d'interface est plus simple à apprendre et à utiliser pour les utilisateurs néophytes. Cette application web a été développée grâce à des technologies modernes. En effet, sa conception repose principalement sur un *framework* JavaScript qui supporte ECMAScript 6 et les versions supérieures de ce standard (ES6+). Celui-ci a été développé spécifiquement pour les besoins de cette application. Par ailleurs, la stratégie de déploiement qui a été adoptée pour ce composant applicatif demeure la même que pour l'API, c'est-à-dire, un service répliqué admettant deux instances qui peuvent être placées soit sur un nœud *manager*, soit sur un *worker*.

La Figure 6.3 montre une capture d'écran de la vue principale de l'interface qui permet de créer des *jobs* d'apprentissage machine. Le fil d'Ariane (`A`) permet d'indiquer aux utilisateurs

41. https://github.com/FlorentinTh/LE2ML-GUI

les étapes qui doivent être complétées pour définir toutes les tâches du *pipeline*. Dans une vue précédente à cette capture d'écran, l'utilisateur doit sélectionner le type d'apprentissage machine qu'il souhaite réaliser (dans le cas présent, il s'agit d'un apprentissage machine traditionnel). Pour chaque étape du *pipeline*, un ensemble de composants web (B) est proposé pour permettre aux utilisateurs de configurer chaque tâche de manière pratique. Cependant, bien qu'il soit possible de construire les fichiers de définition en interagissant avec l'interface graphique, une fonction d'import (C) a également été prévue pour permettre la réutilisation des définitions de *pipeline* existantes. Dans ce contexte, les composants web pour chacune des étapes sont préremplis en fonction du contenu du fichier téléchargé.

Par ailleurs, cette interface offre également aux utilisateurs un moyen d'interagir avec toutes les autres fonctionnalités de l'API. Le menu *Data* (d) regroupe toutes les fonctionnalités liées aux fichiers de données d'entrée. Celles-ci comprennent un composant pour téléverser de nouveaux fichiers, une vue permettant de naviguer dans le système de fichiers distribué, un outil de visualisation des données ainsi qu'un accès aux différents modules de prétraitement et de fusion des données. Le menu *Jobs* (e) présente la liste des *jobs* d'apprentissage machine en cours de traitement et terminés qui ont été créés par l'utilisateur connecté. Cette liste est mise à jour en temps réel grâce à la technologie Server-Sent Events (SSE) qui permet à l'API d'envoyer des notifications *push* à l'application web. Par le biais de cette vue, il est également possible de télécharger les fichiers résultant d'un *job* complété. Ces fichiers peuvent être soit

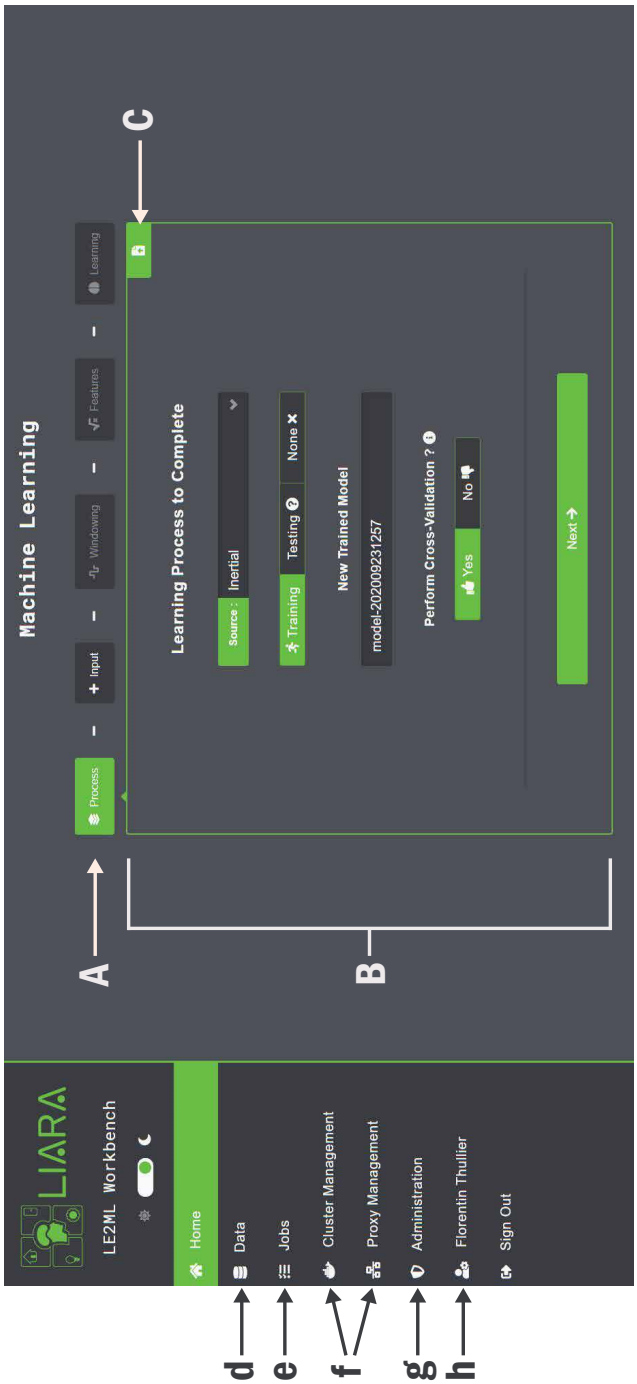


Figure 6.3 : Capture d'écran de l'application web qui montre la première étape de la définition d'un *pipeline* d'apprentissage machine.

un rapport comprenant des mesures d'évaluation de la performance qui sont calculées pendant la phase d'entraînement, soit la liste des prédictions déterminées lors de la phase de test. Ensuite, les menus *Cluster Management* et *Proxy Management* (`f`) redirigent vers les tableaux de bord de Portainer et de Træfik respectivement. Le menu *Administration* (`g`) permet aux utilisateurs qui ont le rôle administrateur de gérer les modules, les clés d'application ainsi que les utilisateurs inscrits. Enfin, les utilisateurs connectés ont également accès à leur profil (`i`), où il leur est possible de modifier la plupart de leurs informations personnelles, y compris leur mot de passe.

6.1.3 MODULES PROPOSÉS

Dans la mesure où LE2ML, de par sa conception, a pour objectif de fournir une meilleure flexibilité, plusieurs modules ont été conçus pour réaliser certaines tâches spécifiques qui composent le processus d'apprentissage machine. Leur développement a été simplifié afin que quiconque soit capable d'étendre les fonctionnalités du *workbench* en proposant ses propres mécanismes. Ainsi, il est possible de proposer des modules pour le traitement de données telles que les méthodes de prétraitement ou de fusion de données, mais également, des modules qui proposent de nouvelles techniques d'extraction de caractéristiques ou de nouveaux algorithmes d'apprentissage machine.

Le premier avantage principal qui motive l'usage de modules est qu'ils fournissent un moyen simple de pallier la diversité dans la conception de logiciels, tant en termes de langages de programmation, que d'environnements d'exécution qui forment le vaste monde des

techniques liées au processus d'apprentissage machine. Par exemple, lorsque des chercheurs développent de nouveaux algorithmes, ces derniers ne devraient pas avoir à se préoccuper de la compatibilité de leur nouvelle implémentation avec un environnement existant. Ainsi, grâce à un outil modulaire, qui demeure agnostique et flexible, ils ont la possibilité d'évaluer les performances de leur méthode en utilisant la technologie qu'ils jugent la plus appropriée, tout en bénéficiant de la puissance des outils dont elle dispose (*p. ex.* librairies, *framework*, *etc.*). Le deuxième avantage à l'utilisation de modules est qu'ils facilitent considérablement l'exécution en parallèle et la mise à l'échelle de nombreux processus, puisque ceux-ci s'exécutent au sein de conteneurs. Bien qu'il soit possible d'imaginer une quantité infinie de modules, cette section n'en décrit que trois qui ont été développés spécifiquement pour répondre aux besoins de notre laboratoire. Plus précisément, ces modules doivent permettre la réplication de la méthode de reconnaissance des types de sols proposée au Chapitre 4.

FENÊTRAGE

Nous avons vu que le processus d'apprentissage machine peut impliquer l'utilisation de signaux ou de séries temporelles comme données d'entrée. Par conséquent, il est parfois nécessaire d'appliquer une fonction de fenêtre glissante pour segmenter ces données brutes en de plus petits ensembles. En ce sens, puisque plusieurs travaux proposés par notre laboratoire de recherche se sont basés sur l'exploitation de telles données (Thullier *et al.*, 2017; Chapron *et al.*, 2018; Bouchard *et al.*, 2020), nous avons décidé de fournir un module de fenêtrage

pour LE2ML [42]. Développé en JavaScript ES6+, ce module propose les fonctions de fenêtrage les plus connues qui sont généralement appliquées sur ce type de données, c'est-à-dire les fonctions rectangulaires, triangulaires, de Hamming (cf. Équation 2.2), de Hann (cf. Équation 2.4) et de Blackmann (cf. Équation 2.5). En outre, ce module permet également de configurer plusieurs autres paramètres dont ce processus de segmentation dépend, comme la taille de la fenêtre glissante (cf. ligne 14 de la Figure 6.2). Ce paramètre s'exprime dans différentes unités selon que les données admettent un horodatage (*timestamp*) ou non. Par exemple, si les données ne sont pas horodatées, la taille de la fenêtre peut être donnée en fonction d'une unité de fréquence (*p. ex.* Hz), sinon une unité de temps (*p. ex.* secondes ou minutes) peut être préférée. De plus, il est également possible de définir un paramètre de chevauchement qui est alors exprimé en pourcentage de la taille de la fenêtre (cf. ligne 18 de la Figure 6.2). Enfin, dans le but de garantir la meilleure efficacité possible, tant en termes d'utilisation des ressources de calcul que de consommation mémoire, l'implémentation de ce module repose principalement sur un traitement des données en flux (*streams*), qui a été mis en place grâce à l'environnement Node.js.

EXTRACTION DE CARACTÉRISTIQUES

L'extraction de caractéristiques demeure une tâche importante à accomplir dans le traitement du processus d'apprentissage machine. Pour rappel, l'objectif d'une telle tâche est de calculer des vecteurs de caractéristiques discriminantes basés sur chacun des segments

42. https://github.com/FlorentinTh/LE2ML-Windowing-Module

de données brutes obtenus à partir du processus de fenêtrage précédent. En ce sens, puisque plusieurs recherches menées au sein de notre laboratoire ont utilisé des centrales inertielles (IMUs) (Thullier *et al.*, 2017, 2018; Chapron *et al.*, 2018), un module qui propose les principales caractéristiques qu'il est possible d'obtenir grâce à ce type de données a été développé [43]. De ce fait, onze caractéristiques différentes provenant des domaines temporel et fréquentiel peuvent être calculées. Celles-ci incluent l'asymétrie (cf. Équation 2.6), l'aplatissement (cf. Équation 2.7), la composante continue, l'énergie spectrale (cf. Équation 2.9) et l'entropie (cf. Équation 2.10). Par ailleurs, ce module a également été développé en JavaScript `ES6+`. Cependant, bien que ce langage ne soit pas fortement typé et ne semble pas adapté au calcul de formules mathématiques complexes, une précision et des performances identiques ont été observées par rapport au programme écrit en Go [44] utilisé dans la méthode proposée dans le Chapitre 4. En effet, de la même manière que pour le module de fenêtrage, une attention particulière a été portée sur l'efficacité de la consommation de la mémoire lors du développement de ce module, notamment grâce au traitement des fichiers par flux de données. Enfin, cette implémentation se veut également flexible quant au type de signal qu'il reçoit en entrée, car celui-ci s'appuie sur l'utilisation de l'algorithme de la FFT de Bluestein (Bluestein, 1970) pour le passage du domaine temporel au domaine fréquentiel.

La première version des fichiers de définition pour les *pipelines* d'apprentissage machine permet, dans un premier temps, de lister un ensemble de caractéristiques à extraire sur des

43. https://github.com/FlorentinTh/LE2ML-FeatureExtractor-Module

44. https://github.com/LIARALab/GoLang-FeatureExtractor

données brutes fournies en entrée (cf. ligne 22 de la Figure 6.2). Cette liste est soit vide, soit elle contient des objets qui définissent chacune des caractéristiques. Ceux-ci sont composés du libellé et du nom du conteneur (l'environnement d'exécution du module) qui est utilisé pour son calcul. Lorsque cette liste est vide, le processus d'extraction des caractéristiques n'est pas réalisé. Néanmoins, lorsque la liste contient des caractéristiques, leurs définitions peuvent inclure des conteneurs différents. Ainsi, il est possible d'exploiter plusieurs modules de ce type pour compléter le processus d'extraction de caractéristiques sur un même signal. Par conséquent, chaque module est responsable du traitement des caractéristiques qui lui sont associées. Les conteneurs s'exécutent donc en parallèle et ils sont indépendants les uns des autres. De ce fait, la tâche est complétée lorsque le dernier conteneur termine son exécution. Son état est alors mis à jour dans la base de données et une opération de fusion est déclenchée par l'API. Celle-ci permet alors d'assembler chacun des fichiers qui ont été produits par les conteneurs afin qu'un seul fichier résultant soit fourni comme entrée à l'algorithme d'apprentissage machine.

APPRENTISSAGE MACHINE

Le module d'apprentissage machine proposé [45] pour cette première version du *workbench* a été conçu pour permettre l'utilisation d'algorithmes d'apprentissage machine traditionnels uniquement. Celui-ci a été développé en Python 3, car sa conception constitue une surcouche logicielle de la librairie scikit-learn qui est également développée en Python.

45. https://github.com/FlorentinTh/LE2ML-Learning-Module

Ainsi, l'objectif de ce module est de traduire automatiquement les paramètres des fichiers de configuration de *pipelines* (cf. ligne 30 de la Figure 6.2) en code compatible avec l'API de cette librairie. En ce sens, il est possible, dans un premier temps, d'entraîner des modèles grâce à plusieurs algorithmes qui sont définis dans la surcouche, comme *k*-Nearest Neighbors et *random forest*. Ensuite, une évaluation pour cette phase d'entraînement peut être réalisée grâce à la méthode de la validation croisée en 10-plis. Par conséquent, une matrice de confusion ainsi que des mesures pertinentes telles que la justesse, la *F-mesure* et la Kappa de Cohen peuvent être fournies, une fois le *pipeline* complété. Dans un second temps, puisque ce module d'apprentissage machine permet de sauvegarder les modèles d'apprentissage sur le système de fichier distribué, ceux-ci peuvent être réutilisés lors de la phase de reconnaissance afin de produire un ensemble de prédictions. Par ailleurs, les phases d'entraînement et de reconnaissance ont été explicitement configurées pour exploiter, de manière automatique, toutes les ressources disponibles allouées aux conteneurs de ce module.

6.2 EXPÉRIMENTATIONS & RÉSULTATS

Cette section présente une expérimentation qui a été réalisée afin de montrer les capacités non limitatives du *workbench* pour une utilisation académique des *wearable devices* au sein des habitats intelligents. Ainsi, grâce à LE2ML, nous avons été en mesure de reproduire le processus d'apprentissage de la méthode de reconnaissance des types de sols qui a été proposée au Chapitre 4. Pour ce faire, aucun code n'a dû être réécrit ou adapté—seul un fichier de définition de *pipeline* accompagné des données brutes récoltées précédemment ont été

nécessaires. En ce sens, ce sont les données produites par l'IMU 9-axes du *wearable device* qui ont été utilisées. De plus, afin de respecter la méthode qui a été proposée, deux *jobs* distincts ont été créés *via* le *workbench*, chacun correspondant à une définition spécifique pour les différents algorithmes d'apprentissage. La première concerne l'algorithme k-NN configuré avec les paramètres qui ont été identifiés comme optimaux c'est-à-dire, $k = 1$ voisin et une recherche linéaire de celui-ci grâce au calcul de la mesure de distance de Manhattan (D_m). Par ailleurs, la seconde configuration a permis de spécifier les paramètres pour l'algorithme *random forest* soit, $B = 300$ arbres, $F = \lfloor \log_2(m) + 1 \rfloor$ variables aléatoires et l'évaluation du gain en information selon l'entropie de Shannon. De la même manière que dans le Chapitre 4, l'évaluation de la performance a été réalisée grâce à la validation croisée en 10 plis. Les résultats qui ont été obtenus pour cette expérimentation sont présentés dans le Tableau 6.1.

Tableau 6.1 : Évaluations des performances de la reconnaissance des types de sols sur des données inertielles obtenues respectivement avec la méthode originelle et avec LE2ML.

Méthode Originelle			
	Justesse	*F-mesure*	*Kappa de Cohen*
***k*-NN**	0.93	0.93	0.89
Random Forest	0.92	0.92	0.88
***Pieplines* LE2ML**			
	Justesse	*F-mesure*	*Kappa de Cohen*
***k*-NN**	0.91	0.91	0.86
Random Forest	0.92	0.92	0.88

En ce qui concerne ces résultats, il nous est possible d'affirmer que notre *workbench* offre un niveau de précision similaire à celui de WEKA. En effet, les performances de reconnaissance qui ont été obtenues dans les deux cas demeurent pratiquement identiques, et ce, pour les trois métriques d'évaluation. Néanmoins, le faible écart observé avec l'algorithme *k*-NN n'est pas assez significatif et peut s'expliquer simplement par les différences qui résident dans l'implémentation des deux outils.

6.3 CONCLUSION

Dans ce chapitre, un nouveau type de *workbench* d'apprentissage machine appelé LE2ML a été présenté. Grâce à une conception modulaire qui repose sur l'utilisation de microservices, cet outil a été avant tout prévu pour fonctionner au sein de l'architecture distribuée introduite au chapitre précédent. Toutefois, il est également possible que ce *workbench* soit utilisé dans des environnements différents qui admettent, par exemple, une architecture monolithique plus conventionnelle. L'implémentation de LE2ML repose sur une API REST. Cette dernière représente le composant principal du *workbench* puisqu'elle est responsable de l'orchestration de l'ensemble du système et plus particulièrement des modules. Afin de répondre aux besoins exprimés par la définition du processus d'apprentissage, ces modules peuvent être soit réutilisés, soit développés spécifiquement. Ainsi, puisque plusieurs recherches réalisées au LIARA se sont basées sur l'exploitation d'IMUs embarqués dans des *wearable devices*, tous les modules qui ont été présentés dans ce chapitre ont été principalement conçus pour traiter ce type de données, à l'exception du module d'apprentissage machine.

De plus, pour interagir avec l'API, une interface web a été développée, essentiellement pour simplifier la création de fichiers de définition de pipeline qui décrivent l'ensemble du processus d'apprentissage de la machine à réaliser. Ce fichier de définition demeure un élément important du système puisqu'il permet à l'API de piloter des composants logiciels hétérogènes qui font que LE2ML est un *workbench* et agnostique vis-à-vis des technologies. À titre d'exemple, une telle flexibilité a pu être démontrée grâce au *pipeline* d'exemple qui a fait intervenir plusieurs composants développés en JavaScript et en Go, et ce, sans avoir à écrire de code.

Enfin, une des expérimentations réalisées dans le Chapitre 4 a été reproduite *via* ce *workbench*. Les résultats qui ont été obtenus n'ont pas montré de différences significatives en ce qui concerne la performance de reconnaissance par rapport à ceux de la méthode précédente. Par conséquent, malgré un nombre encore limité de modules disponibles, nous pensons que LE2ML demeure un outil indispensable pour favoriser l'intégration des *wearable devices* dans les habitats intelligents en milieu académique—étant donné que cet outil permet de faire le lien entre la couche matérielle offerte par l'architecture et les composants logiciels qui sont développés à la fois pour les *wearable devices*, mais également pour les capteurs ambiants.

CHAPITRE VII

CONCLUSION GÉNÉRALE

Le projet de recherche présenté dans cette thèse propose, de manière originale, de nouveaux travaux qui permettent d'améliorer le contexte de recherche actuel de la reconnaissance d'activités au sein de maisons intelligentes. En effet, les solutions existantes décrites dans le Chapitre 2 ont démontré certaines faiblesses quant à la fiabilité de ces habitats, la plupart du temps, induit par le manque de flexibilité des architectures centralisées qui ne sont plus adaptées aux besoins actuels. Paradoxalement, le domaine des maisons intelligentes a connu, au cours de la dernière décennie, un nombre important d'innovations notamment grâce à l'avènement de l'IoT. Par conséquent, de nouveaux types de capteurs, les *wearable devices*, ont alors vu le jour afin de traiter de manière différente certaines problématiques inhérentes à la reconnaissance d'activités. Cependant, selon la diversité technologique de ces capteurs exposée dans le Chapitre 3, les *wearable devices* souffrent d'une intégration limitée au sein des habitats intelligents existants illustrant de nouveau, un problème majeur de flexibilité dans la conception de des architectures existantes.

Afin de répondre à cette problématique, cette thèse s'est découpée en trois étapes spécifiques. La première a présenté une nouvelle méthode de reconnaissance pour améliorer l'assistance des résidents des maisons intelligentes grâce à un *wearble device*. Cette étape a été nécessaire, car elle nous a permis d'identifier les différents besoins essentiels à une meilleure intégration de ces dispositifs au sein des habitats intelligents. Il est donc apparu que

ceux-ci étaient davantage liés aux composants logiciels des *wearble device* plus qu'au matériel. Ainsi, dans un second temps, nous nous sommes penchés sur la problématique du manque de flexibilité des architectures de ce type d'habitat en proposant une nouvelle architecture capable de proposer un meilleur niveau de fiabilité ainsi qu'une flexibilité permettant de faire cohabiter les différentes applications de la reconnaissance d'activités. Enfin, la dernière phase a permis de lier le tout en introduisant une plateforme d'apprentissage machine permettant la mise en œuvre de processus de reconnaissance d'activités génériques c'est-à-dire, de la même façon, indépendamment du type de capteurs sur lesquels ceux-ci s'appuient.

La suite de ce chapitre va donc revenir sur la réalisation des objectifs, les limitations et les perspectives d'amélioration des travaux proposés ainsi que les apports de ce travail de recherche d'un point de vue personnel.

7.1 RÉALISATION DES OBJECTIFS

Dans le cadre de cette thèse, l'objectif principal consistait à simplifier la mise en place de nouveaux cas d'application pour les *wearable devices* au sein des habitats intelligents. Pour ce faire, il a été, dans un premier temps, nécessaire d'acquérir les connaissances en ce qui concerne la conception de ce type d'habitats. Par conséquent, le Chapitre 2 de cette thèse a présenté les quatre grands types d'architectures de maisons intelligentes existantes en identifiant leurs principaux inconvénients. Par la suite, nous nous sommes intéressés, plus précisément, aux différents processus qui composent la reconnaissance d'activités. Ensuite, le Chapitre 3 nous a permis de mieux cibler la composition des *wearable devices* en détaillant les

différents capteurs que ceux-ci peuvent embarquer, mais également les communications sans fil et les méthodes d'échange de données qui sont utilisées dans leur mise en place actuelle au sein des habitats intelligents.

Dans un second temps, nous avons identifié que les *wearable devices* ont souvent été utilisés dans de nombreux domaines tels que la reconnaissance de gestes et d'activités, la réhabilitation ainsi que pour la surveillance de la santé au sein des habitats intelligents (Khan *et al.*, 2016; Davis *et al.*, 2016; Chapron *et al.*, 2018). Cependant, certaines pistes permettant d'améliorer l'assistance des résidents d'habitats intelligents semblaient pourtant encore inexplorées. Par conséquent, nous avons proposé, dans le Chapitre 4, un nouveau *wearable device* qui permet de reconnaître les différents types de sols. La méthode proposée a donc fait intervenir plusieurs algorithmes et technologies qui ont été présentés dans les chapitres précédents. Grâce à cette contribution, nous avons montré qu'une telle méthode était réalisable et qu'en plus d'offrir un excellent niveau de précision, celle-ci s'inscrit parfaitement dans l'optique d'améliorer l'assistance apportée au résidents de maisons intelligentes. En effet, dans ces habitats, leurs occupants peuvent faire face à différents types de sols. Par conséquent, dans le cas de personnes en perte d'autonomie, ces sols peuvent alors représenter des dangers ou causer de la peur chez les résidents, car ceux-ci demeurent plus ou moins meubles ou glissants. Grâce à un tel cas pratique, nous avons pu déterminer de façon plus précise les problèmes de flexibilité vis-à-vis de l'intégration des *wearable devices* dans les habitats intelligents et plus spécifiquement, dans l'architecture industrielle mise en œuvre au LIARA.

Ainsi, pour répondre à cette problématique d'intégration, la contribution présentée au Chapitre 5 a introduit un nouveau type d'architecture d'habitat intelligent. Cette architecture, inspirée des infrastructures qu'il est possible de retrouver auprès des fournisseurs de services *cloud*, demeure un système distribué qui repose sur l'utilisation de microservices. Par conséquent, elle s'inscrit directement dans la continuité du travail proposé par Plantevin *et al.* (2018) qui constitue les prémices des architectures distribuées fiables et évolutives dans le contexte des habitats intelligents. Cette architecture a permis de montrer à travers différentes expérimentations que sa conception permettait à la fois d'offrir un niveau de fiabilité adéquat pour la sécurité des résidents de maisons intelligentes, ainsi qu'un niveau de flexibilité et d'évolutivité permettant de simplifier le déploiement de nouveaux composants applicatifs et d'améliorer l'interopérabilité de leur exécution.

Enfin, la dernière contribution de cette thèse et qui est présentée dans le Chapitre 6 concerne la mise en œuvre d'un *workbench* d'apprentissage machine modulaire. Bien que ce dernier ait été conçu pour être utilisé de pair avec l'architecture distribuée proposée dans le Chapitre 5, celui demeure exploitable sur n'importe quel type d'architecture sans contraintes particulièrement complexes de déploiement. Cet outil permet de faire le lien entre l'architecture sur laquelle il est déployé et les nouvelles applications des *wearable devices* au sein des habitats intelligents. En effet, puisque ce *workbench* s'appuie sur l'utilisation de modules dont l'exécution est conteneurisée dans des microservices, il permet aux chercheurs de réutiliser les composants logiciels disponibles pour réaliser les différents processus d'apprentissage pour la reconnaissance d'activités. Ainsi, il est possible de déployer aussi bien des applications néces-

saires aux méthodes de reconnaissance qui utilisent aussi bien des *wearable devices* que des capteurs ambiants. De plus, l'aspect modulaire de l'outil proposé permet aux expérimentateurs d'utiliser les langages de programmation et des environnements d'exécutions avec lesquels ils sont les plus familiers ou qui demeurent les plus appropriés pour chaque application, et ce, sans impact sur le fonctionnement final du *workbench*.

7.2 LIMITATIONS ET PERSPECTIVES D'AMÉLIORATION

Bien que chacune des contributions proposées dans cette thèse apporte des réponses aux problématiques qui ont été identifiées dans le premier chapitre, il subsiste certaines limitations qui suggèrent, par conséquent, de possibles voies d'améliorations.

Dans un premier temps, en ce qui concerne le *wearable device* introduit pour permettre la reconnaissance des types de sols, nous pensons que malgré la qualité des résultats obtenus, cette méthode pourrait être évaluée de façon plus approfondie en faisant intervenir un plus grand nombre de personnes. En effet, la principale limitation de cette méthode concerne le faible nombre de participants qui ont été impliqués dans les expérimentations.

Par ailleurs, nous pensons également que l'architecture distribuée mériterait d'être déployée de manière plus professionnelle. Pour ce faire, du matériel plus adapté et plus fiable que les Raspberry Pi devra être utilisé. Celui-ci pourra inclure de vrais serveurs plus puissants, des systèmes de stockage RAID éprouvés, une installation réseau redondante dédiée et des alimentations sans interruption. Cette adaptation aura donc pour objectif principal d'offrir un

environnement adéquat pour le déploiement des futurs application qui seront développées au LIARA.

Enfin, en ce qui concerne le *workbench*, nous projetons, dans un futur proche, de continuer son développement. En effet, bien que les principales fonctionnalités nécessaires à son fonctionnement soient terminées, un nombre limité de modules a cependant été proposé. Toutefois, afin que celui-ci puisse répondre au plus grand nombre de problématiques faisant intervenir un processus d'apprentissage machine, il est envisagé de proposer plusieurs nouveaux modules dans un avenir proche. Ceux-ci devront permettre le support de différents types de données et l'intégration de processus additionnels, tels que des techniques de prétraitement et de fusion des données. Ces nouveaux modules pourraient aussi intégrer de nouvelles fonctionnalités relatives aux technologies de communications sans fil, tel que le support du protocole WebSocket qui n'est pas encore abouti. De ce fait, l'intégration des *wearables devices* au sein des habitats intelligents s'en retrouverait encore améliorée. En outre, une autre considération sera portée sur la conception du *pipeline* d'apprentissage profond qui n'a pas encore été étudiée. Finalement, cet outil devra, de la même manière que pour l'architecture, être déployé dans notre laboratoire de sorte qu'il puisse être utilisé régulièrement par les différents étudiants et chercheurs.

7.3 APPORTS PERSONNELS

L'achèvement de ce doctorat constitue le fruit d'un long travail où rigueur, créativité, autonomie, patience et détermination ont été les maîtres mots pour y parvenir. De ce fait, ce

projet m'a permis de renforcer mon expertise dans divers domaines tels que l'intelligence artificielle, la conception de dispositifs électroniques ou encore les architectures distribuées. De plus, grâce aux travaux qui ont été mis en œuvre tout au long de mon cheminement doctoral, j'ai pu perfectionner mes compétences en anglais et développer une meilleure culture scientifique de manière générale.

Pour finir, la réalisation de ce projet de recherche me motive, aujourd'hui, à poursuivre dans cette voie universitaire aussi bien pour l'aspect recherche en elle-même que pour ce qui concerne l'enseignement.

Mes derniers mots vont aux personnes qui ont été présentes tout au long de ces quatre longues années de doctorat. Je remercie encore une fois les professeurs Sébastien Gaboury et Sylvain Hallé, les personnes du département d'informatique et de mathématique de l'Université du Québec, mes collègues du LIARA. J'en profite également pour remercier le département informatique de l'Institut Universitaire de Technologie de La Rochelle sans qui je n'aurais peut-être jamais réalisé un tel parcours.

BIBLIOGRAPHIE

Abreu, P. H., Xavier, J., Castro Silva, D., Reis, L. P. et Petry, M. (2014). Using Kalman filters to reduce noise from RFID location system. *The Scientific World Journal*. http://dx.doi.org/10.1155/2014/796279

Acampora, G., Cook, D. J., Rashidi, P. et Vasilakos, A. V. (2013). A survey on ambient intelligence in healthcare. *Proceedings of the IEEE*, *101*(12), 2470–2494. http://dx.doi.org/10.1109/JPROC.2013.2262913

Adib, F., Mao, H., Kabelac, Z., Katabi, D. et Miller, R. C. (2015). Smart Homes that Monitor Breathing and Heart Rate. Dans *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems - CHI '15*, 837–846., Seoul, Republic of Korea. ACM. http://dx.doi.org/10.1145/2702123.2702200

Alemdar, H., Durmaz Incel, O., Ertan, H. et Ersoy, C. (2013). ARAS Human Activity Datasets in Multiple Homes with Multiple Residents. Dans *Proceedings of the ICTs for improving Patients Rehabilitation Research Techniques*. http://dx.doi.org/10.4108/icst.pervasivehealth.2013.252120

Altun, K. et Barshan, B. (2010). Human Activity Recognition Using Inertial / Magnetic Sensor Units. *Human Behavior Understanding*. http://dx.doi.org/10.1007/978-3-642-14715-9{_}5

Alzheimer's Association (2018). 2018 Alzheimer's Disease Facts and Figures. *Alzheimer's & Dementia*. http://dx.doi.org/10.1016/j.jalz.2018.02.001

Anguita, D., Ghio, A., Oneto, L., Parra, X. et Reyes-Ortiz, J. L. (2012). Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine. Dans *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. http://dx.doi.org/10.1007/978-3-642-35395-6_30

Anliker, U., Ward, J., Lukowicz, P., Troster, G., Dolveck, F., Baer, M., Keita, F., Schenker, E., Catarsi, F., Coluccini, L., Belardinelli, A., Shklarski, D., Alon, M., Hirt, E., Schmid, R. et Vuskovic, M. (2004). AMON : A Wearable Multiparameter Medical Monitoring

and Alert System. *IEEE Transactions on Information Technology in Biomedicine*, *8*(4), 415–427. http://dx.doi.org/10.1109/TITB.2004.837888

Arlot, S. et Celisse, A. (2010). A survey of cross-validation procedures for model selection. *Statistics Surveys*, *4*, 40–79. http://dx.doi.org/10.1214/09-SS054

Asuncion, A. et Newman, D. J. (2007). UCI Machine Learning Repository : Data Sets.

Ayuningtyas, C., Leitner, G., Hitz, M., Funk, M., Hu, J. et Rauterberg, M. (2014). Activity monitoring for multi-inhabitant smart homes. *SPIE Newsroom*. http://dx.doi.org/10.1117/2.1201412.005697

Bae, J. et Tomizuka, M. (2013). A tele-monitoring system for gait rehabilitation with an inertial measurement unit and a shoe-type ground reaction force sensor. *Mechatronics*. http://dx.doi.org/10.1016/j.mechatronics.2013.06.007

Bamberg, S. J., Benbasat, A. Y., Scarborough, D. M., Krebs, D. E. et Paradiso, J. A. (2008). Gait analysis using a shoe-integrated wireless sensor system. *IEEE Transactions on Information Technology in Biomedicine*. http://dx.doi.org/10.1109/TITB.2007.899493

Banos, O., Galvez, J.-M., Damas, M., Pomares, H. et Rojas, I. (2014). Window size impact in human activity recognition. *Sensors (Basel, Switzerland)*. http://dx.doi.org/10.3390/s140406474

Bao, L. et Intille, S. S. (2004). Activity Recognition from User-Annotated Acceleration Data. In *Pervasive Computing* 1–17.

Bayat, A., Pomplun, M. et Tran, D. A. (2014). A Study on Human Activity Recognition Using Accelerometer Data from Smartphones. *Procedia Computer Science*, *34*, 450–457. http://dx.doi.org/10.1016/J.PROCS.2014.07.009

Belley, C., Gaboury, S., Bouchard, B. et Bouzouane, A. (2014). An efficient and inexpensive method for activity recognition within a smart home based on load signatures of appliances. Dans *Pervasive and Mobile Computing*. http://dx.doi.org/10.1016/j.pmcj.2013.02.002

Bellman, R. E. (1957). *Dynamic Programming*. Princeton University Press.

Ben-David, A. (2007). A lot of randomness is hiding in accuracy. *Engineering Applications of Artificial Intelligence*. http://dx.doi.org/10.1016/j.engappai.2007.01.001

Benatti, S., Casamassima, F., Milosevic, B., Farella, E., Schonle, P., Fateh, S., Burger, T., Huang, Q. et Benini, L. (2015). A Versatile Embedded Platform for EMG Acquisition and Gesture Recognition. *IEEE Transactions on Biomedical Circuits and Systems*, *9*(5), 620–630. http://dx.doi.org/10.1109/TBCAS.2015.2476555

Bergmann, O., Hillmann, K. T. et Gerdes, S. (2012). A CoAP-gateway for smart homes. Dans *2012 International Conference on Computing, Networking and Communications, ICNC'12*, 446–450., Maui, HI, USA. IEEE. http://dx.doi.org/10.1109/ICCNC.2012.6167461

Bibuli, M., Caccia, M. et Lapierre, L. (2007). Path-Following Algorithms and Experiments for an Autonomous Surface Vehicle. *IFAC Proceedings Volumes*, *40*(17), 81–86. http://dx.doi.org/10.3182/20070919-3-HR-3904.00015

Bluestein, L. I. (1970). A Linear Filtering Approach to the Computation of Discrete Fourier Transform. *IEEE Transactions on Audio and Electroacoustics*, *18*(4), 451–455. http://dx.doi.org/10.1109/TAU.1970.1162132

Bluetooth, S. (2017). Mesh Networking Specifications. Récupéré le 2018-02-10 de https://bit.ly/2P4O0M2

Boger, J., Hoey, J., Poupart, P., Boutilier, C., Fernie, G. et Mihailidis, A. (2006). A planning system based on Markov decision processes to guide people with dementia through activities of daily living. *IEEE transactions on information technology in biomedicine : a publication of the IEEE Engineering in Medicine and Biology Society*. http://dx.doi.org/10.1109/TITB.2006.864480

Bouchard, B., Giroux, S. et Bouzouane, A. (2007). A keyhole plan recognition model for alzheimer's patients : First results. *Applied Artificial Intelligence*. http://dx.doi.org/10.1080/08839510701492579

Bouchard, K., Bouchard, B. et Bouzouane, A. (2014). Practical Guidelines to Build Smart Homes : Lessons Learned. In T. &. F. CRC press (dir.), *Opportunistic networking, smart home, smart city, smart systems*, numéro January 2015 1–38.

Bouchard, K., Maitre, J., Bertuglia, C. et Gaboury, S. (2020). Activity Recognition in Smart Homes using UWB Radars. Dans *Procedia Computer Science*, volume 170, 10–17. http://dx.doi.org/10.1016/j.procs.2020.03.004

Bouckaert, R. R., Frank, E., Hall, M. A., Holmes, G., Pfahringer, B., Reutemann, P. et Witten, I. H. (2010). WEKA - Experiences with a java open-source project. *Journal of Machine Learning Research*, *11*, 2533–2541.

Breiman, L. (2001). Random Forests. *Machine Learning*, *45*(1), 5–32. http://dx.doi.org/https://doi.org/10.1023/A:1010933404324

Brigham, E. O. et Morrow, R. E. (1967). The fast Fourier transform. *IEEE Spectrum*. http://dx.doi.org/10.1109/MSPEC.1967.5217220

Brumitt, B., Meyers, B., Krumm, J., Kern, A. et Shafer, S. (2000). Easyliving : Technologies for intelligent environments. Dans *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. http://dx.doi.org/10.1007/3-540-39959-3_2

Buettner, M., Prasad, R., Philipose, M. et Wetherall, D. (2009). Recognizing daily activities with RFID-based sensors. Dans *Proceedings of the 11th international conference on Ubiquitous computing - Ubicomp '09*. http://dx.doi.org/10.1145/1620545.1620553

Butterworth, S. (1930). On the theory of filter amplifiers. http://dx.doi.org/citeulike-article-id:5322726

Chang, Y. S., Hung, Y. S., Chang, C. L. et Juang, T. Y. (2009). Toward a NFC phone-driven context awareness smart environment. Dans *UIC-ATC 2009 - Symposia and Workshops on Ubiquitous, Autonomic and Trusted Computing in Conjunction with the UIC'09 and ATC'09 Conferences*, Brisbane, QLD, Australia. IEEE. http://dx.doi.org/10.1109/UIC-ATC.2009.37

Chapelle, O., Schölkopf, B., Chapelle, Chapelle, O., Schölkopf, B. et Rai, P. (2006). *Semi-*

Supervised Learning. http://dx.doi.org/10.1007/s12539-009-0016-2

Chapron, K., Plantevin, V., Thullier, F., Bouchard, K., Duchesne, E. et Gaboury, S. (2018). A More Efficient Transportable and Scalable System for Real-Time Activities and Exercises Recognition. *Sensors*, *18*(1), 268. http://dx.doi.org/10.3390/s18010268

Chen, Y., Zhao, Z., Wang, S. et Chen, Z. (2012). Extreme learning machine-based device displacement free activity recognition model. *Soft Computing*. http://dx.doi.org/10.1007/s00500-012-0822-8

Cheng, S. T., Wang, C. H. et Horng, G. J. (2012). OSGi-based smart home architecture for heterogeneous network. *Expert Systems with Applications*. http://dx.doi.org/10.1016/j.eswa.2012.04.077

Cho, W. T., Lai, Y. X., Lai, C. F. et Huang, Y. M. (2013). Appliance-aware activity recognition mechanism for iot energy management system. *Computer Journal*, *56*(8), 1020–1033. http://dx.doi.org/10.1093/comjnl/bxt047

Cleland, I., Kikhia, B., Nugent, C., Boytsov, A., Hallberg, J., Synnes, K., McClean, S. et Finlay, D. (2013). Optimal placement of accelerometers for the detection of everyday activities. *Sensors (Basel, Switzerland)*. http://dx.doi.org/10.3390/s130709183

Cook, D. J., Crandall, A. S., Thomas, B. L. et Krishnan, N. C. (2013). CASAS : A smart home in a box. *Computer*, *46*(7), 62–69. http://dx.doi.org/10.1109/MC.2012.328

Cook, D. J. et Schmitter-Edgecombe, M. (2009). Assessing the quality of activities in a smart environment. Dans *Methods of Information in Medicine*. http://dx.doi.org/10.3414/ME0592

Cook, D. J., Youngblood, M., Heierman, E. O., Gopalratnam, K., Rao, S., Litvin, A. et Khawaja, F. (2003). MavHome : An agent-based smart home. *Proceedings of the 1st IEEE International Conference on Pervasive Computing and Communications, PerCom 2003*, 521–524. http://dx.doi.org/10.1109/percom.2003.1192783

Crandall, A. S. et Cook, D. J. (2009). Coping with multiple residents in a smart environment. *Journal of Ambient Intelligence and Smart Environments*. http://dx.doi.org/10.3233/

Crea, S., De Rossi, S. M., Donati, M., Reberšek, P., Novak, D., Vitiello, N., Lenzi, T., Podobnik, J., Munih, M. et Carrozza, M. C. (2012). Development of gait segmentation methods for wearable foot pressure sensors. Dans *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*. http://dx.doi.org/10.1109/EMBC.2012.6347120

Cruz, F. R. G., Sejera, M. P., Bunnao, M. B. G., Jovellanos, B. R., Maano, P. L. C. et Santos, C. J. R. (2018). Fall detection wearable device interconnected through ZigBee network. Dans *HNICEM 2017 - 9th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management*, 1–6. IEEE. http://dx.doi.org/10.1109/HNICEM.2017.8269563

Cubo, J., Nieto, A. et Pimentel, E. (2014). A cloud-based internet of things platform for ambient assisted living. *Sensors*, *14*(8), 14070–14105. http://dx.doi.org/10.3390/s140814070

Davis, K., Owusu, E., Bastani, V., Marcenaro, L., Hu, J., Regazzoni, C. et Feijs, L. (2016). Activity recognition based on inertial sensors for Ambient Assisted Living. Dans *19th International Conference on Information Fusion (FUSION)*, 371–378., Heidelberg.

Delachaux, B., Rebetez, J., Perez-Uribe, A. et Satizábal Mejia, H. F. (2013). Indoor activity recognition by combining One-vs.-All neural network classifiers exploiting wearable and depth sensors. Dans *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. http://dx.doi.org/10.1007/978-3-642-38682-4_25

Delahoz, Y. S. et Labrador, M. A. (2014). Survey on fall detection and fall prevention using wearable and external sensors. http://dx.doi.org/10.3390/s141019806

Demšar, J., Curk, T., Erjavec, A., Gorup, Č., Hočevar, T., Milutinovič, M., Možina, M., Polajnar, M., Toplak, M., Starič, A., Štajdohar, M., Umek, L., Žagar, L., Žbontar, J., Žitnik, M. et Zupan, B. (2013). Orange : Data mining toolbox in python. *Journal of Machine Learning Research*, *14*(1), 2349–2353. http://dx.doi.org/10.5555/2567709.2567736

Demšar, J., Zupan, B., Leban, G. et Curk, T. (2004). Orange : From experimental machine

learning to interactive data mining. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *3202*, 537–539. http://dx.doi.org/10.1007/978-3-540-30116-5_58

Díaz-Rodríguez, N., Grönroos, S., Wickström, F., Lilius, J., Eertink, H., Braun, A., Dillen, P., Crowley, J. et Alexandersson, J. (2018). An ontology for wearables data interoperability and ambient assisted living application development. In *Studies in Fuzziness and Soft Computing* 559–568. Springer International Publishing

Dikaiakos, M. D., Katsaros, D., Mehra, P., Pallis, G. et Vakali, A. (2009). Cloud computing : Distributed internet computing for IT and scientific research. *IEEE Internet Computing*, *13*(5), 10–11. http://dx.doi.org/10.1109/MIC.2009.103

Dossot, D. (2014). *RabbitMQ Essentials*. Packt Publishing.

Dragoni, N., Giallorenzo, S., Lafuente, A. L., Mazzara, M., Montesi, F., Mustafin, R. et Safina, L. (2017). Microservices : Yesterday, today, and tomorrow. In M. Mazzara et B. Meyer (dir.), *Present and Ulterior Software Engineering* 195–216. Springer

Emi, I. A. et Stankovic, J. A. (2015). SARRIMA : Smart ADL Recognizer and Resident Identifier in Multi-resident Accommodations. *Proceedings of the Conference on Wireless Health*. http://dx.doi.org/10.1145/2811780.2811916

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters*. http://dx.doi.org/10.1016/j.patrec.2005.10.010

Fielding, R. (2000). *Architectural Styles and the Design of Network-based Software Architectures*. (Thèse de doctorat). University of California, Irvine. http://dx.doi.org/10.1.1.91.2433

Fielding, R. et Reschke, J. (2014). Hypertext Transfer Protocol (HTTP/1.1) : Message Syntax and Routing. http://dx.doi.org/10.17487/RFC7230. Récupéré de https://www.rfc-editor.org/info/rfc7230

Figo, D., Diniz, P. C., Ferreira, D. R. et Cardoso, J. M. P. (2010). Preprocessing techniques for context recognition from accelerometer data. *Personal and Ubiquitous Computing*.

http://dx.doi.org/10.1007/s00779-010-0293-9

Fortin-Simard, D., Bilodeau, J.-S., Bouchard, K., Gaboury, S., Bouchard, B. et Bouzouane, A. (2015). Exploiting Passive RFID Technology for Activity Recognition in Smart Homes. *IEEE Intelligent Systems*. http://dx.doi.org/10.1109/MIS.2015.18

Friedman, N., Geiger, D. et Goldszmit, M. (1997). Bayesian Network Classifiers. *Machine Learning*. http://dx.doi.org/10.1023/a:1007465528199

Gan, J. et Tao, Y. (2015). DBSCAN revisited : Mis-claim, un-fixability, and approximation. Dans *Proceedings of the ACM SIGMOD International Conference on Management of Data*, volume 2015-May, 519–530., Melbourne, Victoria, Australia. http://dx.doi.org/10.1145/2723372.2737792

Gao, L., Bourke, A. et Nelson, J. (2014). Evaluation of accelerometer based multi-sensor versus single-sensor activity recognition systems. *Medical Engineering & Physics*, *36*(6), 779–785. http://dx.doi.org/10.1016/J.MEDENGPHY.2014.02.012

Garcia-Ceja, E., Brena, R., Carrasco-Jimenez, J., Garrido, L., Garcia-Ceja, E., Brena, R. F., Carrasco-Jimenez, J. C. et Garrido, L. (2014). Long-Term Activity Recognition from Wristwatch Accelerometer Data. *Sensors*, *14*(12), 22500–22524. http://dx.doi.org/10.3390/s141222500

Gaskin, J., Jenkins, J., Meservy, T., Steffen, J. et Payne, K. (2017). Using Wearable Devices for Non-invasive, Inexpensive Physiological Data Collection. *Proceedings of the 50th Hawaii International Conference on System Sciences*. http://dx.doi.org/10.24251/HICSS.2017.072

Ghaffarinejad, A. et Syrotiuk, V. R. (2014). Load balancing in a campus network using software defined networking. Dans *Proceedings - 2014 3rd GENI Research and Educational Experiment Workshop, GREE 2014*, 75–76., Atlanta, GA, USA. IEEE. http://dx.doi.org/10.1109/GREE.2014.9

Ghayvat, H., Mukhopadhyay, S., Shenjie, B., Chouhan, A. et Chen, W. (2018). Smart home based ambient assisted living : Recognition of anomaly in the activity of daily living for an elderly living alone. Dans *I2MTC 2018 - 2018 IEEE International Instrumentation and Measurement Technology Conference : Discovering New Horizons in Instrumentation and*

Measurement, Proceedings, 1–5. IEEE. http://dx.doi.org/10.1109/I2MTC.2018.8409885

Ghazvininejad, M., Rabiee, H. R., Pourdamghani, N. et Khanipour, P. (2011). HMM based semi-supervised learning for activity recognition. Dans *Proceedings of the 2011 international workshop on Situation activity & goal awareness - SAGAware '11*. http://dx.doi.org/10.1145/2030045.2030065

Giovannetti, T., Libon, D. J., Buxbaum, L. J. et Schwartz, M. F. (2002). Naturalistic action impairments in dementia. *Neuropsychologia*. http://dx.doi.org/10.1016/S0028-3932(01)00229-9

Giroux, S., Leblanc, T., Bouzouane, A., Bouchard, B., Pigot, H. et Bauchet, J. (2009). The Praxis of Cognitive Assistance in Smart Homes. *BMI Book*, 183–211. http://dx.doi.org/10.3233/978-1-60750-048-3-183

Godfrey, A., Hetherington, V., Shum, H., Bonato, P., Lovell, N. H. et Stuart, S. (2018). From A to Z : Wearable technology explained. *Maturitas*, *113*, 40–47. http://dx.doi.org/10.1016/j.maturitas.2018.04.012

Gomez, C., Oller, J. et Paradells, J. (2012). Overview and evaluation of bluetooth low energy : An emerging low-power wireless technology. *Sensors*, *12*(9), 11734–11753. http://dx.doi.org/10.3390/s120911734

Guenterberg, E., Ostadabbas, S., Ghasemzadeh, H. et Jafari, R. (2009). An Automatic Segmentation Technique in Body Sensor Networks based on Signal Energy. Dans *Proceedings of the 4th International ICST Conference on Body Area Networks*, p. 21. ICST. http://dx.doi.org/10.4108/ICST.BODYNETS2009.6036

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P. et Witten, I. H. (2009). The WEKA data mining software : An update. *ACM SIGKDD Explorations Newsletter*, *11*(1), 10–18. http://dx.doi.org/10.1145/1656274.1656278

Handa, A., Sharma, A. et Shukla, S. K. (2019). Machine learning in cybersecurity : A review. *Wiley Interdisciplinary Reviews : Data Mining and Knowledge Discovery*, *9*(4). http://dx.doi.org/10.1002/widm.1306

Harris, C. et Hunter, S. (2016). Smart-home technologies were found to support some domains of independent living when ageing at home : Perspectives of older adult consumers' families, health professionals and service providers. *Australian Occupational Therapy Journal*, *63*(6), 439–440. http://dx.doi.org/10.1111/1440-1630.12323

Haux, R., Koch, S., Lovell, N., Marschollek, M., Nakashima, N. et Wolf, K.-H. (2016). Health-Enabling and Ambient Assistive Technologies : Past, Present, Future. *Yearbook of Medical Informatics*, *25*(S 01), S76–S91. http://dx.doi.org/10.15265/IYS-2016-s008

He, Z. et Jin, L. (2009). Activity recognition from acceleration data based on discrete consine transform and SVM. Dans *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics*. http://dx.doi.org/10.1109/ICSMC.2009.5346042

Heckerman, D., Geiger, D. et Chickering, D. M. (1995). Learning Bayesian Networks : The Combination of Knowledge and Statistical Data. *Machine Learning*. http://dx.doi.org/10.1023/A:1022623210503

Helal, S., Mann, W., El-Zabadani, H., King, J., Kaddoura, Y. et Jansen, E. (2005). The Gator tech smart house : A programmable pervasive space. *Computer*, *38*(3), 50–60. http://dx.doi.org/10.1109/MC.2005.107

Hintjens, P. (2013). *ZeroMQ Messaging for Many Applications*. O'Reilly Media.

Hofmann, M. et Klinkenberg, R. (2014). *RapidMiner : Data Mining Use Cases and Business Analytics Applications*. CRC Press.

Holmes, G., Donkin, A. et Witten, I. H. (1994). WEKA : A machine learning workbench. Dans *Australian and New Zealand Conference on Intelligent Information Systems - Proceedings*, 357–361. http://dx.doi.org/10.1109/anziis.1994.396988

Hornik, K., Buchta, C. et Zeileis, A. (2009). Open-source machine learning : R meets Weka. *Computational Statistics*, *24*(2), 225–232. http://dx.doi.org/10.1007/s00180-008-0119-7

Hu, Y., Tilke, D., Adams, T., Crandall, A. S., Cook, D. J. et Schmitter-Edgecombe, M. (2016). Smart home in a box : usability study for a large scale self-installation of smart home technologies. *Journal of Reliable Intelligent Environments*. http://dx.doi.org/10.1007/

s40860-016-0021-y

Huang, T. S., Yang, G. J. et Tang, G. Y. (1979). A Fast Two-Dimensional Median Filtering Algorithm. *IEEE Transactions on Acoustics, Speech, and Signal Processing*. http://dx.doi.org/10.1109/TASSP.1979.1163188

Hui, T. K., Sherratt, R. S. et Sánchez, D. D. (2017). Major requirements for building Smart Homes in Smart Cities based on Internet of Things technologies. *Future Generation Computer Systems*, *76*, 358–369. http://dx.doi.org/10.1016/j.future.2016.10.026

Huifeng, W., Kadry, S. N. et Raj, E. D. (2020). Continuous health monitoring of sportsperson using IoT devices based wearable technology. *Computer Communications*, *160*, 588–595. http://dx.doi.org/10.1016/j.comcom.2020.04.025

Hunkeler, U., Truong, H. L. et Stanford-Clark, A. (2008). MQTT-S - A publish/subscribe protocol for wireless sensor networks. Dans *3rd IEEE/Create-Net International Conference on Communication System Software and Middleware, COMSWARE*, 791–798., Bangalore, India. IEEE. http://dx.doi.org/10.1109/COMSWA.2008.4554519. Récupéré de http://ieeexplore.ieee.org/document/4554519/

Huynh, T. et Schiele, B. (2005). Analyzing features for activity recognition. Dans *Proceedings of the 2005 joint conference on Smart objects and ambient intelligence innovative context-aware services : usages and technologies - sOc-EUSAI '05*. http://dx.doi.org/10.1145/1107548.1107591

Inomata, T., Naya, F., Kuwahara, N., Hattori, F. et Kogure, K. (2009). Activity Recognition from Interactions with Objects Using Dynamic Bayesian Network. Dans *Proceedings of the 3rd ACM International Workshop on Context-Awareness for Self-Managing Systems*. http://dx.doi.org/10.1145/1538864.1538871

Institute of Electrical and Electronics Engineers. (1999). *IEEE Std 1451.1-1999, IEEE Standard for a Smart Transducer Interface for Sensors and Actuators — Network Capable Application Processor (NCAP) Information Model*. http://dx.doi.org/10.1109/IEEESTD.2000.91313

International Telecommunication Union (2012). *Overview of the Internet of things*. Rapport technique, Geneva, Switzerland.

Istepanian, R. S. H., Hu, S., Philip, N. Y. et Sungoor, A. (2011). The potential of Internet of m-health Things “m-IoT” for non-invasive glucose level sensing. Dans *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 5264–5266., Boston, MA, USA. IEEE. http://dx.doi.org/10.1109/IEMBS.2011.6091302

Jafarnejad Ghomi, E., Masoud Rahmani, A. et Nasih Qader, N. (2017). Load-balancing algorithms in cloud computing : A survey. *Journal of Network and Computer Applications*, *88*, 50–71. http://dx.doi.org/10.1016/j.jnca.2017.04.007

Johnson, I. et Ianes, P. (2018). Frail Elderly Persons and Smart Home Technologies. In S. Masiero et U. Carraro (dir.), *Rehabilitation Medicine for Elderly Patients* 119–123. Cham : Springer International Publishing

Jung, P.-G., Lim, G., Kim, S. et Kong, K. (2015). A Wearable Gesture Recognition Device for Detecting Muscular Activities Based on Air-Pressure Sensors. *IEEE Transactions on Industrial Informatics*, *11*(2), 485–494. http://dx.doi.org/10.1109/TII.2015.2405413

Katz, S., Ford, A. B., Moskowitz, R. W., Jackson, B. A. et Jaffe, M. W. (1963). Studies of Illness in the Aged : The Index of ADL : A Standardized Measure of Biological and Psychosocial Function. *JAMA : The Journal of the American Medical Association*. http://dx.doi.org/10.1001/jama.1963.03060120024016

Kertesz, C. (2016). Rigidity-Based Surface Recognition for a Domestic Legged Robot. *IEEE Robotics and Automation Letters*, *1*(1), 309–315. http://dx.doi.org/10.1109/LRA.2016.2519949

Khan, A. M. (2011). Human Activity Recognition Using A Single Tri-axial Accelerometer. *Computer Engineering*. http://dx.doi.org/10.1587/transfun.E93.A.1379

Khan, Y., Ostfeld, A. E., Lochner, C. M., Pierre, A. et Arias, A. C. (2016). Monitoring of Vital Signs with Flexible and Wearable Medical Devices. *Advanced Materials*, *28*(22), 4373–4395. http://dx.doi.org/10.1002/adma.201504366

Kohavi, R. (1995). A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. Dans *IJCAI’95 Proceedings of the 14th international joint conference on Artificial intelligence*, volume 2, 1137–1143., Montreal, QC, Canada. Morgan Kaufmann Publishers Inc.

Kovashka, A. et Grauman, K. (2010). Learning a hierarchy of discriminative space-time neighborhood features for human action recognition. Dans *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. http://dx.doi.org/10.1109/CVPR.2010.5539881

Kovatsch, M., Duquennoy, S. et Dunkels, A. (2011). A low-power CoAP for Contiki. Dans *8th IEEE International Conference on Mobile Ad-hoc and Sensor Systems, MASS 2011*, 855–860., Valencia, Spain. IEEE. http://dx.doi.org/10.1109/MASS.2011.100

Lago, P., Lang, F., Roncancio, C., Jiménez-Guarín, C., Mateescu, R. et Bonnefond, N. (2017). The contextact@A4H real-life dataset of daily-living activities activity recognition using model checking. Dans *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 10257 LNAI, 175–188. http://dx.doi.org/10.1007/978-3-319-57837-8_14

Langlois, R. E. et Lu, H. (2008). Intelligible machine learning with malibu. Dans *Proceedings of the 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS'08 - "Personalized Healthcare through Technology"*, 3795–3798. http://dx.doi.org/10.1109/iembs.2008.4650035

Larrañaga, P., Calvo, B., Santana, R., Bielza, C., Galdiano, J., Inza, I., Lozano, J. A., Armañanzas, R., Santafé, G., Pérez, A. et Robles, V. (2006). Machine learning in bioinformatics. *Briefings in Bioinformatics*, *7*(1), 86–112. http://dx.doi.org/10.1093/bib/bbk007

Lawton, M. P. et Brody, E. M. (1969). Assessment of Older People : Self-Maintaining and Instrumental Activities of Daily Living. *The Gerontologist*. http://dx.doi.org/10.1093/geront/9.3_Part_1.179

Lee, C., Zappaterra, L., Choi, K. et Choi, H. A. (2014). Securing smart home : Technologies, security challenges, and security requirements. Dans *2014 IEEE Conference on Communications and Network Security*, 67–72., San Francisco, CA, USA. IEEE. http://dx.doi.org/10.1109/CNS.2014.6997467

Leightley, D., Darby, J. et McPhee, J. S. (2013). Human Activity Recognition for Physical Rehabilitation. *2013 IEEE International Conference on Systems, Man, and Cybernetics*. http://dx.doi.org/10.1109/SMC.2013.51

Li, X., Zhang, Y., Marsic, I., Sarcevic, A. et Burd, R. S. (2016). Deep Learning for RFID-Based Activity Recognition. Dans *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM - SenSys '16*. http://dx.doi.org/10.1145/2994551.2994569

MacKenzie, C. M., Laskey, K., McCabe, F., Brown, P. F. et Metz, R. (2006). Reference Model for Service Oriented Architecture 1.0. OASIS Standard. *OASIS Open*, *12*(October), 1–31.

Mahalingam, M., Dutt, D. G., Duda, K., Agarwal, P., Kreeger, L., Sridhar, T., Bursell, M. et Wright, C. (2014). *Virtual eXtensible Local Area Network (VXLAN) : A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks*. RFC 7348, RFC Editor

Mainetti, L., Mighali, V. et Patrono, L. (2015). An IoT-based user-centric ecosystem for heterogeneous Smart Home environments. Dans *2015 IEEE International Conference on Communications (ICC)*, 704–709., London, UK. IEEE. http://dx.doi.org/10.1109/ICC.2015.7248404

Mallat, S. G. (1989). A Theory for Multiresolution Signal Decomposition : The Wavelet Representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. http://dx.doi.org/10.1109/34.192463

Mannini, A., Rosenberger, M., Haskell, W. L., Sabatini, A. M. et Intille, S. S. (2017). Activity recognition in youth using single accelerometer placed at wrist or ankle. *Medicine and Science in Sports and Exercise*. http://dx.doi.org/10.1249/MSS.0000000000001144

Martin, J. (2014). Bluetooth Smart's Rise From Obscurity to Mainstream. Récupéré le 2018-10-02 de https://ubm.io/2NVafaN

Maurer, U., Smailagic, A., Siewiorek, D. P. et Deisher, M. (2006). Activity recognition and monitoring using multiple sensors on different body positions. Dans *International Workshop on Wearable and Implantable Body Sensor Networks (BSN'06)*, 113–116., Cambridge, MA, USA. IEEE. http://dx.doi.org/10.1109/BSN.2006.6

Messing, R., Pal, C. et Kautz, H. (2009). Activity recognition using the velocity histories of tracked keypoints. Dans *Proceedings of the IEEE International Conference on Computer Vision*. http://dx.doi.org/10.1109/ICCV.2009.5459154

Mihailidis, A., Barbenel, J. C. et Fernie, G. (2004). The efficacy of an intelligent cognitive orthosis to facilitate handwashing by persons with moderate to severe dementia. http://dx.doi.org/10.1080/09602010343000156

Mitchell, E., Monaghan, D. et O'Connor, N. E. (2013). Classification of sporting activities using smartphone accelerometers. *Sensors (Switzerland)*. http://dx.doi.org/10.3390/s130405317

Mokhtari, G., Anvari-Moghaddam, A., Zhang, Q. et Karunanithi, M. (2018). Multi-residential activity labelling in smart homes with wearable tags using BLE technology. *Sensors (Switzerland)*. http://dx.doi.org/10.3390/s18030908

Mukhopadhyay, S. C. (2014). Wearable sensors for human activity monitoring : A review. *IEEE Sensors Journal*. http://dx.doi.org/10.1109/JSEN.2014.2370945

Nazerfard, E., Das, B., Holder, L. B. et Cook, D. J. (2010). Conditional random fields for activity recognition in smart environments. Dans *Proceedings of the ACM international conference on Health informatics - IHI '10*, p. 282., New York, New York, USA. ACM Press. http://dx.doi.org/10.1145/1882992.1883032

Nielsen, C. (2014). Tech-Styles : are Consumers Really Interested In Wearing Tech on Their Sleeves ? Récupéré le 2019-09-18 de https://bit.ly/2kQvX2h

Novák, M. et Binas, M. (2011). An architecture overview of the smart-home system based on OSGi. Dans *SCYR 2011 : 11th Scientific Conference of Young Researchers of Faculty of Electrical Engineering and Informatics Technical University of Ko{š}ice*, 221–224.

NPD Group (2015). The Demographic Divide : Fitness Trackers and Smartwatches Attracting Very Different Segments of the Market, According to The NPD Group. Récupéré le 2019-09-20 de https://bit.ly/1IRegAj

Oliver, N., Garg, A. et Horvitz, E. (2004). Layered representations for learning and inferring office activity from multiple sensory channels. Dans *Computer Vision and Image Understanding*. http://dx.doi.org/10.1016/j.cviu.2004.02.004

ON World Inc. (2017). *Bluetooth Low Energy IoT : A Market Dynamics Report*.

Ondrus, J. et Pigneur, Y. (2007). An assessment of NFC for future mobile payment systems. Dans *Conference Proceedings - 6th International Conference on the Management of Mobile Business, ICMB 2007*. IEEE. http://dx.doi.org/10.1109/ICMB.2007.9

Otis, M. J. D., Ayena, J. C., Tremblay, L. E., Fortin, P. E. et Ménélas, B.-A. J. (2016). Use of an Enactive Insole for Reducing the Risk of Falling on Different Types of Soil Using Vibrotactile Cueing for the Elderly. *PLOS ONE*, *11*(9), e0162107. http://dx.doi.org/10.1371/journal.pone.0162107

Paraponaris, A., Davin, B. et Verger, P. (2012). Formal and informal care for disabled elderly living in the community : An appraisal of French care composition and costs. *European Journal of Health Economics*, *13*(3), 327–336. http://dx.doi.org/10.1007/s10198-011-0305-3

Pärkkä, J., Ermes, M., Korpipää, P., Mäntyjärvi, J., Peltola, J. et Korhonen, I. (2006). Activity classification using realistic data from wearable sensors. *IEEE transactions on information technology in biomedicine : a publication of the IEEE Engineering in Medicine and Biology Society*. http://dx.doi.org/10.1109/TITB.2005.856863

Patel, S., Park, H., Bonato, P., Chan, L. et Rodgers, M. (2012). A review of wearable sensors and systems with application in rehabilitation. http://dx.doi.org/10.1186/1743-0003-9-21

Patterson, D. J., Fox, D., Kautz, H. et Philipose, M. (2005). Fine-grained activity recognition by aggregating abstract object usage. Dans *Proceedings - International Symposium on Wearable Computers, ISWC*. http://dx.doi.org/10.1109/ISWC.2005.22

Pering, T., Anokwa, Y. et Want, R. (2007). Gesture connect : Facilitating tangible interaction with a flick of the wrist. Dans *1st International Conference on Tangible and Embedded Interaction*, Baton Rouge, Louisiana, USA. ACM. http://dx.doi.org/10.1145/1226969.1227022

Perumal, T., Ramli, A. R., Leong, C. Y., Mansor, S. et Samsudin, K. (2008). Interoperability among Heterogeneous Systems in Smart Home Environment. Dans *SITIS 2008 - Proceedings of the 4th International Conference on Signal Image Technology and Internet Based Systems*, 177–186., Bali, Indonesia. IEEE. http://dx.doi.org/10.1109/SITIS.2008.94

Plantevin, V. (2018). *Une nouvelle architecture distribuée pour la reconnaissance d'activités au sein d'une maison intelligente*. (Thèse de doctorat). Université du Québec à

Chicoutimi. Récupéré de https://constellation.uqac.ca/4573/

Plantevin, V., Bouzouane, A., Bouchard, B. et Gaboury, S. (2018). Towards a more reliable and scalable architecture for smart home environments. *Journal of Ambient Intelligence and Humanized Computing*. http://dx.doi.org/10.1007/s12652-018-0954-5

Plantevin, V., Bouzouane, A. et Gaboury, S. (2017). The light node communication framework : A new way to communicate inside smart homes. *Sensors*, *17*(10), 2397–2416. http://dx.doi.org/10.3390/s17102397

Prince, M., Comas-Herrera, A., Knapp, M., Guerchet, M. et Karagiannidou, M. (2016). World Alzheimer Report 2016 Improving healthcare for people living with dementia. Coverage, Quality and costs now and in the future. *Alzheimer's Disease International (ADI)*. http://dx.doi.org/10.13140/RG.2.2.22580.04483

Quinlan Ross, J. (1993). C4. 5 : Programs For Machine Learning. http://dx.doi.org/10.1016/S0019-9958(62)90649-6

Rahmani, A. M., Thanigaivelan, N. K., Gia, T. N., Granados, J., Negash, B., Liljeberg, P. et Tenhunen, H. (2015). Smart e-Health Gateway : Bringing intelligence to Internet-of-Things based ubiquitous healthcare systems. Dans *2015 12th Annual IEEE Consumer Communications and Networking Conference, CCNC 2015*, 826–834. IEEE. http://dx.doi.org/10.1109/CCNC.2015.7158084

Rajkomar, A., Dean, J. et Kohane, I. (2019). Machine learning in medicine. *New England Journal of Medicine*, *380*(14), 1347–1358. http://dx.doi.org/10.1056/NEJMra1814259

Ramirez-Prado, G., Barmada, B. et Liesaputra, V. (2019). Non-intrusive behavior awareness for residents of a smart house. Dans *Proceedings - 2019 IEEE International Conference on Big Data, Big Data 2019*, 5269–5273. http://dx.doi.org/10.1109/BigData47090.2019.9005550

Rashidi, P. et Mihailidis, A. (2013). A survey on ambient-assisted living tools for older adults. *IEEE Journal of Biomedical and Health Informatics*. http://dx.doi.org/10.1109/JBHI.2012.2234129

Ravi, N., Dandekar, N., Mysore, P. et Littman, M. M. L. (2005). Activity Recognition from Accelerometer Data. Dans *Proceedings of the 17th Conference on Innovative Applications of Artificial Intelligence*, 1541–1546., Pittsburgh, PA, USA. http://dx.doi.org/10.1007/978-3-642-02481-8_120

Ritthoo, O., Klinkenberg, R., Fischer, S., Mierswa, I. et Felske, S. (2003). *Yale : Yet Another Learning Environment*. Rapport technique, Universität Dortmund, Dortmund

Rogers, W. A., Meyer, B., Walker, N. et Fisk, A. D. (1998). Functional Limitations to Daily Living Tasks in the Aged : A Focus Group Analysis. *Human Factors : The Journal of the Human Factors and Ergonomics Society*. http://dx.doi.org/10.1518/001872098779480613

Roy, P. C., Bouchard, B., Bouzouane, A. et Giroux, S. (2013). Ambient Activity Recognition in Smart Environments for Cognitive Assistance. *International Journal of Robotics Applications and Technologies (IJRAT)*, *1*(1), 29–56. http://dx.doi.org/10.4018/ijrat.2013010103

Russell, S. J. et Norvig, P. (2010). *Artificial Intelligence : A Modern Approach* (3rd éd.). Pearson.

Sadri, F. (2011). Ambient intelligence : A Survey. *ACM Computing Surveys*. http://dx.doi.org/10.1145/1978802.1978815

Saint-Andre, P. (2011). *Extensible Messaging and Presence Protocol (XMPP) : Core*. Rapport technique

Salowey, J., Choudhury, A. et McGrew, D. (2008). *AES Galois Counter Mode (GCM) Cipher Suites for TLS*. RFC 5288, RFC Editor

Sanford, J., Young, C., Cremer, S., Popa, D., Bugnariu, N. et Patterson, R. (2015). Grip Pressure and Wrist Joint Angle Measurement during Activities of Daily Life. *Procedia Manufacturing*. http://dx.doi.org/10.1016/j.promfg.2015.07.321

Sant'Anna, A. et Wickström, N. (2010). A symbol-based approach to gait analysis from acceleration signals : Identification and detection of gait events and a new measure of gait symmetry. *IEEE Transactions on Information Technology in Biomedicine*. http://dx.doi.org/10.1109/TITB.2010.2047402

Schmidt, C. F., Sridharan, N. S. et Goodson, J. L. (1978). The plan recognition problem : An intersection of psychology and artificial intelligence. http://dx.doi.org/10.1016/0004-3702(78)90012-7

Sekine, M., Tamura, T., Togawa, T. et Fukui, Y. (2000). Classification of waist-acceleration signals in a continuous walking record. *Medical Engineering & Physics*, *22*(4), 285–291. http://dx.doi.org/10.1016/S1350-4533(00)00041-2

Seon-Woo Lee et Mase, K. (2002). Activity and location recognition using wearable sensors. *IEEE Pervasive Computing*, *1*(3), 24–32. http://dx.doi.org/10.1109/MPRV.2002.1037719

Shelby, Z. (2010). Embedded web services. *IEEE Wireless Communications*, *17*(6), 52–57. http://dx.doi.org/10.1109/MWC.2010.5675778

Shelby, Z., Hartke, K. et Bormann, C. (2014). The constrained application protocol (CoAP). http://dx.doi.org/10.17487/RFC7252

Shoaib, M., Scholten, H. et Havinga, P. J. (2013). Towards physical activity recognition using smartphone sensors. Dans *2013 IEEE 10th International Conference on Ubiquitous Intelligence and Computing and 2013 IEEE 10th International Conference on Autonomic and Trusted Computing*, 80–87., Vietri sul Mere, Italy. IEEE. http://dx.doi.org/10.1109/UIC-ATC.2013.43

Stikic, M., Huynh, T., Van Laerhoven, K. et Schiele, B. (2008). ADL recognition based on the combination of RFID and accelerometer sensing. *2008 Second International Conference on Pervasive Computing Technologies for Healthcare*. http://dx.doi.org/10.1109/PCTHEALTH.2008.4571084

Sun, W., Choi, M. et Choi, S. (2013). IEEE 802.11 ah : A long range 802.11 WLAN at sub 1 GHz. *Journal of ICT Standardization*. http://dx.doi.org/10.13052/jicts2245-800X

Sysel, P. et Rajmic, P. (2012). Goertzel algorithm generalized to non-integer multiples of fundamental frequency. *Eurasip Journal on Advances in Signal Processing*. http://dx.doi.org/10.1186/1687-6180-2012-56

Tanenbaum, A. S. et Wetherall, D. J. (2010). *Computer Networks* (5th éd.). Upper Saddle

River, NJ, USA : Prentice Hall Press.

Tapia, E. M., Intille, S. S., Haskell, W., Larson, K. W. J., King, A. et Friedman, R. (2007). Real-Time Recognition of Physical Activities and their Intensitiies Using Wireless Accelerometers and a Heart Monitor. *International Symposium on Wearable Computers*. http://dx.doi.org/10.1109/ISWC.2007.4373774

Tavakoli, M., Benussi, C., Alhais Lopes, P., Osorio, L. B. et de Almeida, A. T. (2018). Robust hand gesture recognition with a double channel surface EMG wearable armband and SVM classifier. *Biomedical Signal Processing and Control*, *46*, 121–130. http://dx.doi.org/10.1016/J.BSPC.2018.07.010

Thullier, F., Plantevin, V., Bouzouane, A., Halle, S. et Gaboury, S. (2017). A position-independent method for soil types recognition using inertial data from a wearable device. Dans *2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, 1–10., San Francisco, CA, USA. IEEE. http://dx.doi.org/10.1109/UIC-ATC.2017.8397511

Thullier, F., Plantevin, V., Bouzouane, A., Halle, S. et Gaboury, S. (2018). A Comparison of Inertial Data Acquisition Methods for a Position-Independent Soil Types Recognition. Dans *2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, 1052–1056., Guangzhou, China. IEEE. http://dx.doi.org/10.1109/SmartWorld.2018.00183

Tunca, C., Alemdar, H., Ertan, H., Incel, O. D. et Ersoy, C. (2014). Multimodal wireless sensor network-based ambient assisted living in real homes with multiple residents. *Sensors (Switzerland)*. http://dx.doi.org/10.3390/s140609692

UNFPA (2007). The State of the World Population 2007 - Unleashing The Potential of Urban Growth. *Linking Population, Poverty and Development*. http://dx.doi.org/ISBN978-0-89714-807-8

United Nations. (2015). *World Population Ageing 2015*. http://dx.doi.org/ST/ESA/SER.A/390

United Nations (2017a). The Sustainable Development Goals Report. *United Nations Publications*. http://dx.doi.org/10.18356/3405d09f-en

United Nations (2017b). *World Population Prospects The 2017 Revision Key Findings and Advance Tables*. Rapport technique

Upadhyay, Y., Borole, A. et Dileepan, D. (2016). MQTT based secured home automation system. Dans *2016 Symposium on Colossal Data Analysis and Networking (CDAN)*, 1–4., Indore, India. IEEE. http://dx.doi.org/10.1109/CDAN.2016.7570945

Vacher, M., Istrate, D., Portet, F., Joubert, T., Chevalier, T., Smidtas, S., Meillon, B., Lecouteux, B., Sehili, M., Chahuara, P. et Méniard, S. (2011). The SWEET-HOME project : Audio technology in smart homes to improve well-being and reliance. Dans *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*. http://dx.doi.org/10.1109/IEMBS.2011.6091309

Vail, D. et Veloso, M. (2004). Learning from accelerometer data on a legged robot. *IFAC Proceedings Volumes*, *37*(8), 822–827. http://dx.doi.org/10.1016/S1474-6670(17)32082-7

Van Den Bossche, A., Gonzalez, N., Val, T., Brulin, D., Vella, F., Vigouroux, N. et Campo, E. (2018). Specifying an MQTT Tree for a Connected Smart Home. Dans M. Mokhtari, B. Abdulrazak, et H. Aloulou (dir.). *Smart Homes and Health Telematics, Designing a Better Future : Urban Assisted Living (ICOST)*, 236–246., Cham. Springer International Publishing. http://dx.doi.org/10.1007/978-3-319-94523-1_21

Van Der Maaten, L. J. P., Postma, E. O. et Van Den Herik, H. J. (2009). Dimensionality Reduction : A Comparative Review. *Journal of Machine Learning Research*. http://dx.doi.org/10.1080/13506280444000102

Van Kasteren, T., Noulas, A., Englebienne, G. et Kröse, B. (2008). Accurate activity recognition in a home setting. Dans *Proceedings of the 10th international conference on Ubiquitous computing - UbiComp '08*. http://dx.doi.org/10.1145/1409635.1409637

Van Kasteren, T. L. M., Englebienne, G. et Kröse, B. J. A. (2011). Hierarchical activity recognition using automatically clustered actions. Dans *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. http://dx.doi.org/10.1007/978-3-642-25167-2_9

Vikramaditya Jakkula, D. J. C. (2007). Mining Sensor Data in Smart Environment for Temporal Activity Prediction. *The 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Vinoski, S. (2006). Advanced message queuing protocol. *IEEE Internet Computing*, *10*(6), 87 – 89. http://dx.doi.org/10.1109/MIC.2006.116

Wang, J., Chen, Y., Hao, S., Peng, X. et Hu, L. (2018). Deep learning for sensor-based activity recognition : A Survey. *Pattern Recognition Letters*. http://dx.doi.org/10.1016/j.patrec.2018.02.010

Wang, W. Z., Guo, Y. W., Huang, B. Y., Zhao, G. R., Liu, B. Q. et Wang, L. (2011). Analysis of filtering methods for 3D acceleration signals in body sensor network. Dans *Proceedings of 2011 International Symposium on Bioelectronics and Bioinformatics, ISBB 2011*. http://dx.doi.org/10.1109/ISBB.2011.6107697

Weiser, M. (1991). The Computer for the 21st Century. *Scientific American*. http://dx.doi.org/10.1038/scientificamerican0991-94

Weiss, C., Fechner, N., Stark, M. et Zell, A. (2007). Comparison of different approaches to vibration-based terrain classification. Dans *Proceedings of the 3rd European Conference on Mobile Robots, EMCR 2007*, 1–6., Freiburg, Germany.

Welch, G. et Bishop, G. (2006). An Introduction to the Kalman Filter. *In Practice*. http://dx.doi.org/10.1.1.117.6808

Witten, I. H., Frank, E., Hall, M. A. et Pal, C. J. (2016). *Data Mining : Practical Machine Learning Tools and Techniques* (4 éd.). Morgan Kaufmann Publishers Inc. http://dx.doi.org/10.1016/c2009-0-19715-5

Yang, J. B., Nguyen, M. N., San, P. P., Li, X. L. et Krishnaswamy, S. (2015). Deep convolutional neural networks on multichannel time series for human activity recognition. Dans *IJCAI International Joint Conference on Artificial Intelligence*. http://dx.doi.org/10.3897/zookeys.77.769

Yuan Jie Fan, Yue Hong Yin, Li Da Xu, Yan Zeng et Fan Wu (2014). IoT-Based Smart

Rehabilitation System. *IEEE Transactions on Industrial Informatics*, *10*(2), 1568–1577. http://dx.doi.org/10.1109/TII.2014.2302583

Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S. et Stoica, I. (2010). Spark : Cluster computing with working sets. Dans ACM (dir.). *2nd USENIX Workshop on Hot Topics in Cloud Computing, HotCloud 2010*, 1–10., Boston, MA. USENIX Association. http://dx.doi.org/10.5555/1863103.1863113

Zheng, Y., Peng, Y., Wang, G., Liu, X., Dong, X. et Wang, J. (2016). Development and evaluation of a sensor glove for hand function assessment and preliminary attempts at assessing hand coordination. *Measurement : Journal of the International Measurement Confederation*. http://dx.doi.org/10.1016/j.measurement.2016.06.059

Zhenyu, Z., Ke-Jun, L., Ruzhen, L. et Shaofeng, W. (2011). Smart home system based on ipv6 and zigbee technology. *Procedia Engineering*, *15*, 1529–1533. http://dx.doi.org/10.1016/j.proeng.2011.08.284

Zhihua, S. (2016). Design of smart home system based on ZigBee. Dans *2016 International Conference on Robots & Intelligent System (ICRIS)*, 167–170., ZhangJiaJie, China. IEEE Comput. Soc. http://dx.doi.org/10.1109/ICRIS.2016.35

Zhu, X. (2005). Semi-Supervised Learning Literature Survey. *SciencesNew York*, *10*, 1–60. http://dx.doi.org/10.1.1.146.2352

ANNEXE A

RÉSULTATS DÉTAILLÉS DE L'EXPÉRIMENTATION POUR LA RECONNAISSANCE DES SOLS

A.1 *WEARABLE DEVICE* : VERSION 1

Tableau A.1 : Résultats détaillés de la reconnaissance des sols avec l'algorithme *Random Forest* configuré avec $B = 150$ arbres et de haut en bas les trois valeurs de F : F_0, F_1 et F_2 pour la version 1 du *wearable device*.

Jeu de données	*Justesse*	*F-mesure*	*Kappa de Cohen*
soil_type_wear_v1_6	0.82	0.82	0.76
soil_type_position_wear_v1_6	0.86	0.86	0.85
Moyenne	0.84	0.84	0.81
Médiane	0.84	0.84	0.81
Jeu de données	*Justesse*	*F-mesure*	*Kappa de Cohen*
soil_type_wear_v1_6	0.82	0.81	0.75
soil_type_position_wear_v1_6	0.87	0.87	0.86
Moyenne	0.85	0.84	0.81
Médiane	0.85	0.84	0.81
Jeu de données	*Justesse*	*F-mesure*	*Kappa de Cohen*
soil_type_wear_v1_6	0.82	0.82	0.76
soil_type_position_wear_v1_6	0.87	0.87	0.86
Moyenne	0.85	0.85	0.81
Médiane	0.85	0.85	0.81

Tableau A.2 : Résultats détaillés de la reconnaissance des sols avec l'algorithme *Random Forest* configuré avec $B = 300$ arbres et de haut en bas les trois valeurs de F : F_0, F_1 et F_2 pour la version 1 du *wearable device*.

Jeu de données	*Justesse*	*F-mesure*	*Kappa de Cohen*
soil_type_wear_v1_6	0.87	0.83	0.77
soil_type_position_wear_v1_6	0.88	0.87	0.87
Moyenne	0.88	0.85	0.82
Médiane	0.88	0.85	0.82
Jeu de données	*Justesse*	*F-mesure*	*Kappa de Cohen*
soil_type_wear_v1_6	0.84	0.84	0.78
soil_type_position_wear_v1_6	0.87	0.87	0.86
Moyenne	0.86	0.86	0.82
Médiane	0.86	0.86	0.82
Jeu de données	*Justesse*	*F-mesure*	*Kappa de Cohen*
soil_type_wear_v1_6	0.82	0.82	0.76
soil_type_position_wear_v1_6	0.87	0.87	0.86
Moyenne	0.85	0.85	0.81
Médiane	0.85	0.85	0.81

Tableau A.3 : Résultats détaillés de la reconnaissance des sols avec l'algorithme des Kappa de Cohen plus proches voisins configuré avec $k = 1$ et de haut en bas la distance euclidienne et la distance de Manhattan pour la version 1 du *wearable device*.

Jeu de données	*Justesse*	*F-mesure*	*Kappa de Cohen*
soil_type_wear_v1_6	0.85	0.85	0.80
soil_type_position_wear_v1_6	0.83	0.83	0.82
Moyenne	0.84	0.84	0.81
Médiane	0.84	0.84	0.81
Jeu de données	*Justesse*	*F-mesure*	*Kappa de Cohen*
soil_type_wear_v1_6	0.87	0.87	0.83
soil_type_position_wear_v1_6	0.85	0.85	0.84
Moyenne	0.86	0.86	0.84
Médiane	0.86	0.86	0.84

A.2 *WEARABLE DEVICE* : VERSION 2

Tableau A.4 : Résultats détaillés de la reconnaissance des sols avec l'algorithme *Random Forest* configuré avec $B = 150$ arbres et $F = f0$ pour la version 2 du *wearable device*.

Jeu de données	*Justesse*	*F-mesure*	*Kappa de Cohen*
soil_type_wear_v2_6	0.86	0.86	0.82
soil_type_wear_v2_9	0.91	0.91	0.86
soil_type_wear_v2_12	0.89	0.89	0.83
soil_type_position_wear_v2_6	0.91	0.90	0.90
soil_type_position_wear_v2_9	0.93	0.92	0.92
soil_type_position_wear_v2_12	0.92	0.92	0.91
Moyenne	0.90	0.90	0.87
Médiane	0.91	0.91	0.88

Tableau A.5 : Résultats détaillés de la reconnaissance des sols avec l'algorithme *Random Forest* configuré avec $B = 150$ arbres et $F = f1$ pour la version 2 du *wearable device*.

Jeu de données	*Justesse*	*F-mesure*	*Kappa de Cohen*
soil_type_wear_v2_6	0.86	0.86	0.81
soil_type_wear_v2_9	0.91	0.91	0.87
soil_type_wear_v2_12	0.89	0.89	0.83
soil_type_position_wear_v2_6	0.91	0.91	0.91
soil_type_position_wear_v2_9	0.92	0.92	0.91
soil_type_position_wear_v2_12	0.92	0.92	0.92
Moyenne	0.90	0.90	0.88
Médiane	0.91	0.91	0.89

Tableau A.6 : Résultats détaillés de la reconnaissance des sols avec l'algorithme *Random Forest* configuré avec $B = 150$ arbres et $F = f2$ pour la version 2 du *wearable device*.

Jeu de données	*Justesse*	*F-mesure*	*Kappa de Cohen*
soil_type_wear_v2_6	0.86	0.86	0.81
soil_type_wear_v2_9	0.90	0.90	0.85
soil_type_wear_v2_12	0.90	0.90	0.85
soil_type_position_wear_v2_6	0.91	0.90	0.90
soil_type_position_wear_v2_9	0.92	0.92	0.91
soil_type_position_wear_v2_12	0.91	0.91	0.91
Moyenne	0.90	0.90	0.87
Médiane	0.91	0.90	0.88

Tableau A.7 : Résultats détaillés de la reconnaissance des sols avec l'algorithme *Random Forest* configuré avec $B = 300$ arbres et $F = f0$ pour la version 2 du *wearable device*.

Jeu de données	*Justesse*	*F-mesure*	*Kappa de Cohen*
soil_type_wear_v2_6	0.86	0.86	0.82
soil_type_wear_v2_9	0.92	0.92	0.87
soil_type_wear_v2_12	0.90	0.90	0.85
soil_type_position_wear_v2_6	0.92	0.92	0.92
soil_type_position_wear_v2_9	0.93	0.93	0.92
soil_type_position_wear_v2_12	0.92	0.92	0.91
Moyenne	0.91	0.91	0.88
Médiane	0.92	0.92	0.89

Tableau A.8 : Résultats détaillés de la reconnaissance des sols avec l'algorithme *Random Forest* configuré avec $B = 300$ arbres et $F = f1$ pour la version 2 du *wearable device*.

Jeu de données	*Justesse*	*F-mesure*	*Kappa de Cohen*
soil_type_wear_v2_6	0.87	0.87	0.83
soil_type_wear_v2_9	0.92	0.92	0.88
soil_type_wear_v2_12	0.90	0.90	0.84
soil_type_position_wear_v2_6	0.92	0.92	0.92
soil_type_position_wear_v2_9	0.92	0.92	0.92
soil_type_position_wear_v2_12	0.92	0.92	0.91
Moyenne	0.91	0.91	0.88
Médiane	0.92	0.92	0.90

Tableau A.9 : Résultats détaillés de la reconnaissance des sols avec l'algorithme *Random Forest* configuré avec $B = 300$ arbres et $F = f2$ pour la version 2 du *wearable device*.

Jeu de données	*Justesse*	*F-mesure*	*Kappa de Cohen*
soil_type_wear_v2_6	0.86	0.86	0.82
soil_type_wear_v2_9	0.92	0.92	0.87
soil_type_wear_v2_12	0.91	0.91	0.86
soil_type_position_wear_v2_6	0.92	0.92	0.92
soil_type_position_wear_v2_9	0.92	0.92	0.92
soil_type_position_wear_v2_12	0.92	0.92	0.92
Moyenne	0.91	0.91	0.89
Médiane	0.92	0.92	0.90

Tableau A.10 : Résultats détaillés de la reconnaissance des sols avec l'algorithme des Kappa de Cohen plus proches voisins configuré avec $k = 1$ et la distance euclidienne pour la version 2 du *wearable device*.

Jeu de données	*Justesse*	*F-mesure*	*Kappa de Cohen*
soil_type_wear_v2_6	0.91	0.91	0.87
soil_type_wear_v2_9	0.90	0.90	0.85
soil_type_wear_v2_12	0.84	0.84	0.76
soil_type_position_wear_v2_6	0.84	0.84	0.83
soil_type_position_wear_v2_9	0.88	0.88	0.88
soil_type_position_wear_v2_12	0.83	0.83	0.82
Moyenne	0.87	0.87	0.84
Médiane	0.86	0.86	0.84

Tableau A.11 : Résultats détaillés de la reconnaissance des sols avec l'algorithme des Kappa de Cohen plus proches voisins configuré avec $k = 1$ et la distance de Manhattan pour la version 2 du *wearable device*.

Jeu de données	*Justesse*	*F-mesure*	*Kappa de Cohen*
soil_type_wear_v2_6	0.92	0.92	0.90
soil_type_wear_v2_9	0.93	0.93	0.89
soil_type_wear_v2_12	0.90	0.90	0.85
soil_type_position_wear_v2_6	0.87	0.87	0.86
soil_type_position_wear_v2_9	0.91	0.91	0.91
soil_type_position_wear_v2_12	0.87	0.87	0.86
Moyenne	0.90	0.90	0.88
Médiane	0.91	0.91	0.88

A.3 TÉLÉPHONE INTELLIGENT

Tableau A.12 : Résultats détaillés de la reconnaissance des sols avec l'algorithme *Random Forest* configuré avec $B = 150$ arbres et $F = f0$ pour le téléphone intelligent.

Jeu de données	*Justesse*	*F-mesure*	*Kappa de Cohen*
soil_type_cell_6	0.92	0.92	0.87
soil_type_cell_9	0.92	0.92	0.88
soil_type_cell_12	0.91	0.91	0.87
soil_type_position_cell_6	0.91	0.91	0.91
soil_type_position_cell_9	0.91	0.91	0.91
soil_type_position_cell_12	0.92	0.92	0.91
Moyenne	0.92	0.92	0.89
Médiane	0.92	0.92	0.90

Tableau A.13 : Résultats détaillés de la reconnaissance des sols avec l'algorithme *Random Forest* configuré avec $B = 150$ arbres et $F = f1$ pour le téléphone intelligent.

Jeu de données	*Justesse*	*F-mesure*	*Kappa de Cohen*
soil_type_cell_6	0.92	0.92	0.88
soil_type_cell_9	0.92	0.92	0.88
soil_type_cell_12	0.92	0.92	0.88
soil_type_position_cell_6	0.92	0.92	0.91
soil_type_position_cell_9	0.92	0.92	0.92
soil_type_position_cell_12	0.92	0.92	0.91
Moyenne	0.92	0.92	0.90
Médiane	0.92	0.92	0.90

Tableau A.14 : Résultats détaillés de la reconnaissance des sols avec l'algorithme *Random Forest* configuré avec $B = 150$ arbres et $F = f2$ pour le téléphone intelligent.

Jeu de données	*Justesse*	*F-mesure*	*Kappa de Cohen*
soil_type_cell_6	0.92	0.92	0.89
soil_type_cell_9	0.91	0.91	0.87
soil_type_cell_12	0.92	0.92	0.88
soil_type_position_cell_6	0.92	0.92	0.91
soil_type_position_cell_9	0.92	0.92	0.91
soil_type_position_cell_12	0.92	0.92	0.92
Moyenne	0.92	0.92	0.90
Médiane	0.92	0.92	0.90

Tableau A.15 : Résultats détaillés de la reconnaissance des sols avec l'algorithme *Random Forest* configuré avec $B = 300$ arbres et $F = f0$ pour le téléphone intelligent.

Jeu de données	*Justesse*	*F-mesure*	*Kappa de Cohen*
soil_type_cell_6	0.92	0.92	0.88
soil_type_cell_9	0.92	0.92	0.88
soil_type_cell_12	0.92	0.92	0.88
soil_type_position_cell_6	0.92	0.92	0.91
soil_type_position_cell_9	0.92	0.92	0.92
soil_type_position_cell_12	0.92	0.92	0.91
Moyenne	0.92	0.92	0.90
Médiane	0.92	0.92	0.90

Tableau A.16 : Résultats détaillés de la reconnaissance des sols avec l'algorithme *Random Forest* configuré avec $B = 300$ arbres et $F = f1$ pour le téléphone intelligent.

Jeu de données	*Justesse*	*F-mesure*	*Kappa de Cohen*
soil_type_cell_6	0.92	0.92	0.88
soil_type_cell_9	0.92	0.92	0.88
soil_type_cell_12	0.92	0.92	0.87
soil_type_position_cell_6	0.92	0.92	0.91
soil_type_position_cell_9	0.92	0.92	0.92
soil_type_position_cell_12	0.92	0.92	0.91
Moyenne	0.92	0.92	0.90
Médiane	0.92	0.92	0.90

Tableau A.17 : Résultats détaillés de la reconnaissance des sols avec l'algorithme *Random Forest* configuré avec $B = 300$ arbres et $F = f2$ pour le téléphone intelligent.

Jeu de données	*Justesse*	*F-mesure*	*Kappa de Cohen*
soil_type_cell_6	0.93	0.93	0.89
soil_type_cell_9	0.92	0.92	0.88
soil_type_cell_12	0.92	0.92	0.88
soil_type_position_cell_6	0.92	0.92	0.91
soil_type_position_cell_9	0.92	0.92	0.91
soil_type_position_cell_12	0.92	0.92	0.91
Moyenne	0.92	0.92	0.90
Médiane	0.92	0.92	0.90

Tableau A.18 : Résultats détaillés de la reconnaissance des sols avec l'algorithme des Kappa de Cohen plus proches voisins configuré avec $k = 1$ et la distance euclidienne pour le téléphone intelligent.

Jeu de données	*Justesse*	*F-mesure*	*Kappa de Cohen*
soil_type_cell_6	0.90	0.90	0.85
soil_type_cell_9	0.90	0.90	0.85
soil_type_cell_12	0.90	0.90	0.85
soil_type_position_cell_6	0.89	0.89	0.88
soil_type_position_cell_9	0.88	0.88	0.87
soil_type_position_cell_12	0.89	0.89	0.88
Moyenne	0.89	0.89	0.86
Médiane	0.90	0.90	0.86

Tableau A.19 : Résultats détaillés de la reconnaissance des sols avec l'algorithme des Kappa de Cohen plus proches voisins configuré avec $k = 1$ et la distance de Manhattan pour le téléphone intelligent.

Jeu de données	*Justesse*	*F-mesure*	*Kappa de Cohen*
soil_type_cell_6	0.92	0.92	0.87
soil_type_cell_9	0.92	0.92	0.87
soil_type_cell_12	0.92	0.92	0.89
soil_type_position_cell_6	0.90	0.90	0.89
soil_type_position_cell_9	0.89	0.89	0.88
soil_type_position_cell_12	0.91	0.91	0.90
Moyenne	0.91	0.91	0.88
Médiane	0.92	0.92	0.88