Institut **S**upérieur
d'**I**nformatique, de
Modélisation et de
leurs **A**pplications

Campus des Cézaux
24 avenue des Landais
BP 10125
63173 AUBIERE Cedex

Rapport d'ingénieur
Projet de 3e année

Filière : Informatique des systèmes embarqués

Création d'un réseau d'objets intelligents, génériques et dynamiques : SmartDevCom

---

Thomas Impery
Cyrille Pierre

Tuteur : Michel Cheminat
Référent : Christian Laforest

Soutenance : 10/03/2016
Projet de 120 heures par personne

Résumé

Le but de ce projet de dernière année est de réaliser une **intelligence artificielle** capable de communiquer avec nous, cela dans le but de créer un réseau d'objets intelligents sans limite physique au niveau du support de communication de l'objet, de sa position et enfin de son type. Ce réseau permettra donc de pouvoir parler avec un objet pouvant se trouver à quelques mètres de nous, mais aussi à l'autre bout de la planète. Il nous autorisera de la même manière de pouvoir allumer la lumière de notre maison, mais aussi de pouvoir observer l'évolution de la température de notre piscine. Le dernier but de ce projet est de pouvoir être indépendant de la technologie de communication.

Afin de réaliser ce projet, il faut en premier lieu savoir ce qui caractérise un **objet intelligent**. Après avoir listé les fonctionnalités que peut avoir n'importe quel type d'objet, il faut mettre en place un **protocole** afin de pouvoir communiquer avec eux. La construction de ce protocole passe en premier lieu par la **classification** de l'ensemble des objets que l'on sera à même de pouvoir créer. Ensuite, il faut porter ce protocole sur les objets intelligents. Ces objets sont constitués d'une carte de développement **Mbed** et d'un ou plusieurs capteur(s) et/ou actionneur(s). Le protocole doit aussi être portée sur une application **Android**. Cette application permet d'utiliser la **reconnaissance vocale** d'Android, afin de pouvoir interagir avec les objets autour de nous, de la manière la plus naturelle possible.

Mots clés : Intelligence artificielle, objet intelligent, protocole, classification, Mbed, Android, reconnaissance vocale.

Abstract

The aim of this last year's project, is to realize an **artificial intelligence** able of communicating with us, in order to create a network of smart objects without any physical limit about the communication support, his position and finally his type.

Looking forward to that goal, in the first place, one must know what define a **smart object**. Having listed the features that can have any type of object, one must set a **protocol** to communicate with them. In the first place, the building of this protocol is a **classification** of the whole objects we are able to create. Then, one must brings this protocol on the smart objects. These objects are made of a **Mbed** microchip, and one or several sensors and/or actuators. The protocol must be carried on an **Android** application aswell. This application allows us to use Android **voice recognition**, in order to interact with the objects around us, in the most natural way.

Keywords : Artificial intelligence, smart object, protocol, classification, Mbed, Android, voice recognition.

Table des matières

Table des figures

Liste des tableaux

Introduction

Aujourd'hui les objets connectés sont partout. Dans notre frigo, dans notre portable, et même dans notre brosse à dents. Tous ces systèmes sont capables de parler, mais tous avec un langage différent, impossible à comprendre pour les autres. Tous ces objets ne sont que des entités dispersées, sans intelligence propre, là où normalement nous parlons d'objets intelligents. Tous les constructeurs créent un protocole différent pour faire communiquer leurs propres objets, impliquant une non compabilité avec les objets d'un concurrent, ce qui est préjudiciable pour les gens souhaitant acquérir un panel d'objets de diverses marques. Ainsi il serait dans l'intérêt des utilisateurs d'avoir un moyen de redonner l'intelligence à ses objets, et de réussir à tous les faire communiquer entre eux, peu importe son origine.

De nos jours, Internet est le réseau le plus connu, et un des réseau des plus abouti. Nous parlons d'Internet des objets pour parler des objets connectés au sens large, mais ceux-ci n'utilisent pas encore de protocole qui permettent un acheminement des données tel que Internet peut le faire. Ainsi notre projet se basera sur les travaux qu'Internet a pu apporter, afin de réaliser un protocole similaire, dédié à l'Internet des objets. Cela dans le but de créer un réseau virtuel se basant sur un réseau physique déjà existant, grâce à Internet lui-même, et des technologies tel que le Bluetooth, afin de relier tous nos objets entre eux.

Ainsi vient donc la question de comment réaliser un réseau pareil. Afin de répondre à cela nous verrons ce qui caractérise véritablement un objet intelligent, puis nous verrons comment mettre en place un protocole générique capable de réaliser la communication de ces objets. Pour finir nous verrons la création de ces différents objets, ainsi que de leurs caractéristiques.

Chapitre 1

Automatisation de la maison

1.1 L'intelligence des objets

1.1.1 Définition d'un objet connecté

Aujourd'hui les objets connectés font partis de notre quotidien, et peuvent réaliser énormément d'actions différentes. Globalement on peut dire qu'un objet connecté, ou l'internet des objets, représente l'extension d'Internet à l'ensemble des objets de notre monde physique. Cela permet de donner de l'intelligence à des objets, afin qu'ils puissent nous communiquer différentes informations (**Capteurs**), ou bien exécuter différentes choses (**Actionneurs**). Ces événements peuvent être réalisés manuellement ou automatiquement selon le besoin.

Malgré tout, ces objets ne sont sémantiquement pas connectés entre eux pour plein de raisons différentes. SmartDevCom a pour but de créer un réseau unique où l'ensemble de ces objets peut communiquer avec les autres. Les objets connectés déjà créés utilisent, pour la plupart, un protocole propriétaire, ce qui induit l'impossibilité d'une quelconque interopérabilité avec les objets d'un autre fournisseur. C'est pourquoi les objets connectés que nous utiliserons pour la mise en place de SmartDevCom seront l'œuvre de notre création. Cet objet connecté est constitué de peu d'éléments :

- Un ou plusieurs capteur(s)
- Un ou plusieurs actionneur(s)
- Une source d'énergie
- Un microcontrôleur
- D'une ou plusieurs interface(s) réseau(x)

1.1.2 Les capteurs

Les capteurs sont tous les objets qui permettent d'obtenir une information sur l'environnement. Ces capteurs permettent par exemple de connaître la température de la pièce, de savoir si la lumière est allumée, ou bien si la porte d'entrée est ouverte.

Ces capteurs permettent une interaction simpliste avec l'environnement puisqu'ils permettent seulement d'obtenir une information physique. Ainsi, l'utilisation d'un capteur est très simple, puisqu'il suffit de lui demander d'envoyer la valeur qu'il contient, pour que ce celui-ci nous l'envoie.

1.1.3 Les actionneurs

Les actionneurs sont tous les objets qui permettent de réaliser une action. Ces actionneurs permettent par exemple d'ouvrir les volets, d'allumer la climatisation, ou bien d'allumer la lumière.

Ils permettent une interaction plus prononcée avec l'environnement, comparativement aux capteurs, étant donné qu'ils ont besoin de paramètres pour réaliser l'action voulue. Par exemple si l'on souhaite allumer une lampe intelligente, il faudra préciser l'intensité lumineuse, ainsi que la couleur de celle-ci.

1.2 La communication des objets

1.2.1 Présentation des technologies de communication

Actuellement, il existe de nombreux protocoles réseaux qui permettent la communication entre les différents objets. Ces protocoles établissent des normes, afin de pouvoir dialoguer et se comprendre entre différents acteurs.

Tous ces protocoles ont leurs avantages et leurs inconvénients. Malgré tous ces protocoles, on peut voir qu'il existe principalement deux familles :

Les protocoles très énergivores qui permettent de faire transiter des données avec un débit très élevé. Cette cadence est permise grâce à une fréquence très élevée. L'inconvénient d'une onde dont la fréquence est très élevée est que celle-ci aura plus de mal à traverser des obstacles, et donc ne sera émise que sur une courte distance.

	Bluetooth	Bluetooth LE /Smart	WIFI	WIFI	WIFI (Next generation)	NFC	Zigbee	Z-Wave	ANT+
Specification	802.15.1	802.15.1	802.11g	802.11n	802.11y	NFCIP-1	802.15.4	Z-Wave alliance	ANT
Frequency	2.4 GHz	2.4 GHz	2.4 GHz	2.4GHz / 5 GHz	3.7GHz (US)	13.56 MHz	868 MHz (EU) 915MHz (US) 2.4GHz	868MHz (EU, China, India, Russia,...) 900MHz (North America, Brazil, HongKong, Australia, Japan,...)	2.4GHz
Range indoor (m)	30	10	25	50	50	0.2	30	45	10
Range max (m)	100	50	75	125	5000	0.2	1500	150	30
Data speed max	3 Mbit/s	1 Mbit/s	54 Mbit/s	540 Mbit/s	54 Mbit/s	424 kbit/s	250 kbit/s	100 kbit/s	<100kbit/s
Data speed typ.	2.1Mbit/s	270 kbit/s	25 Mbit/s	200 Mbit/s	23 Mbit/s	2.5kbit/s	150 kbit/s	40 kbit/s	20 kbit/s
Peak current	150 mA	20mA	150 mA	150 mA	-	15 mA	50 mA	20 mA	35 mA
Sleep current	5 mA	1 uA	100 µA	100 µA	-	10 µA	5 µA	2.5µA	1 µA
Battery life	Month	Year	Day	Day	-	Month/Year	Month/Year	Year	Year
Network topologies	Star	Star	Star	Star	Star	Peer to peer only	Star, Tree, Mesh	Star, Tree, Mesh	Star, Tree, Mesh
Typically :	- Headsets - Computer peripherals	- Mobile phones - Sport trackers -eHealth devices - Wireless sensors	-PC (networking) -WLAN	-same as 802.11g with improved performances -Outdoor LAN	-wireless link between hotspot	-transport ticket -secure payment -door opening	-home automation -wireless sensor networks -smart metering	-home automation	-sport trackers -eHealth devices
Official Website Link	https://www.bluetooth.org/en-us	https://www.bluetooth.org/en-us	http://www.wi-fi.org/	http://www.wi-fi.org/	http://www.wi-fi.org/	http://www.nfc-forum.org/home/	http://www.zigbee.org/	http://www.z-wave.com/	http://www.thisisant.com/

FIGURE 1.1 – Les supports de communications les plus connus

Les protocoles peu énergivores qui possèdent des caractéristiques opposées. En effet ils permettent de faire transiter des données avec un faible débit, sur une grande distance grâce à une fréquence bien moindre.

C'est donc la deuxième famille qui est la plus utilisée pour l'internet des objets. Ceci s'explique par le fait qu'ils n'ont pas besoin d'envoyer beaucoup de données, ont besoin d'envoyer le plus loin possible, et surtout d'avoir le plus d'autonomie possible. Les protocoles les plus représentatifs de cette famille, sont le Bluetooth Low Energy (ou BLE), le Zigbee, et Z-Wave. Ces deux derniers sont relativement onéreux et compliqués à mettre en place, c'est pourquoi le BLE est le protocole le plus démocratisé de cette famille.

1.2.2 Les différentes topologies de réseaux

En plus des différents supports de communication, il existe aussi des topologies différentes pour créer un réseau d'objets. La topologie d'un système représente la manière dont celui-ci est organisé et agencé. Ainsi en fonction des besoins, tout comme pour les protocoles, une topologie sera plus adéquate qu'une autre. On parlera de nœud pour parler des différentes machines présentes dans le réseau, et de chemin ou route pour relier les nœuds entre eux.

Etoile

C'est la topologie la plus commune dans les communications. Elle est aussi appelée l'architecture client/serveur dans le sens où il y une machine qui dirige tout le système, et par lequel toutes les données passent. Il s'agit de l'architecture la plus simple, puisqu'avec cette topologie,

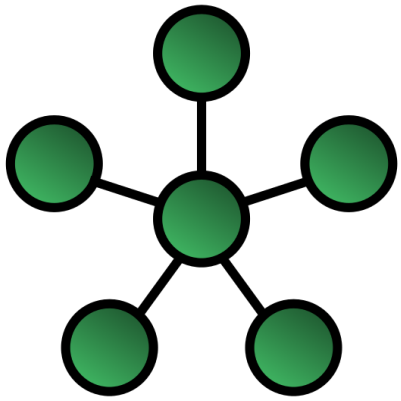


FIGURE 1.2 – Réseau étoile

chaque client se connecte simplement au serveur. Son inconvénient majeur se situe au niveau du nœud principal. En effet si celui-ci ne fonctionne plus, le réseau entier tombe.

P2P

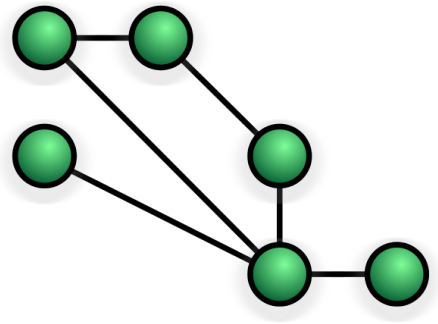


FIGURE 1.3 – Réseau P2P

Cette topologie, aussi appelé réseau maillé, permet d'avoir une architecture réseau plus robuste, qu'une topologie étoile, puisqu'il existe plusieurs chemins pour arriver à un nœud. Ainsi, si un des nœuds ne fonctionnent plus, il ne perturbe que très peu la communication des autres. Elle donne aussi une plus grande sécurité au niveau de l'acheminement des paquets, puisque toutes les données ne passent pas au même endroit, il est donc plus difficile de les sniffer. Pour finir, cette architecture permet d'avoir un réseau beaucoup plus grand et dynamique puisqu'il n'existe pas de nœud central. L'inconvénient de cette architecture est qu'elle est plus dure à mettre en place, du fait qu'il existe plusieurs routes pour acheminer les données. De ce fait, il faut gérer la duplication possible des paquets, et le routage dynamique si un des nœuds n'est plus fonctionnel.

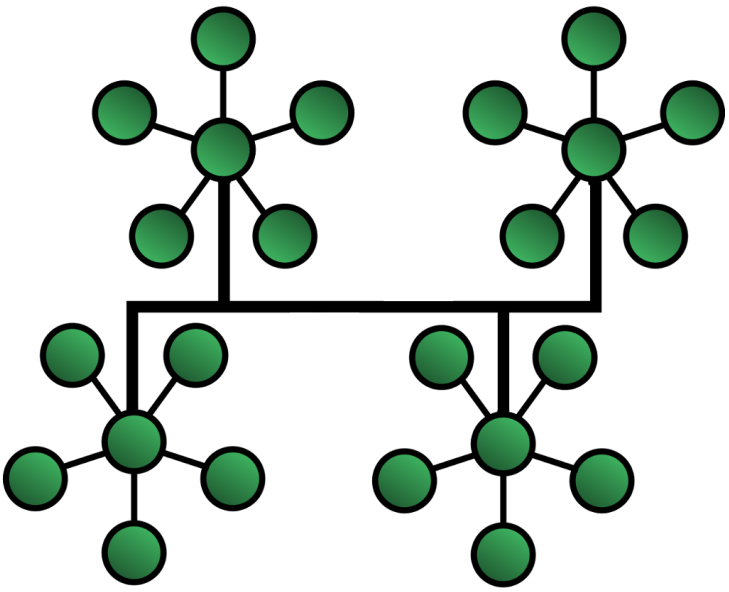


FIGURE 1.4 – Réseau en arbre

Arbre

Cette architecture est simplement l'extension du réseau étoile, c'est l'architecture utilisée par Internet. Il s'agit simplement de la connexion de plusieurs réseaux étoiles entre eux.

1.2.3 Application du réseau à la domotique

Afin de pouvoir être intelligent, les objets doivent communiquer, et donc utiliser un protocole précis. L'ensemble de ces objets doit suivre une topologie, afin de pouvoir acheminer les données correctement. La topologie la plus utilisée pour communiquer avec eux est le P2P. Cela permet de pouvoir communiquer directement avec eux en étant à côté, et cela permet aussi de pouvoir créer un réseau d'objets à taille variable et dynamique.

1.3 La centralisation de l'intelligence : Jarvis

1.3.1 Vision macroscopique d'un système domotique

Actuellement, la croissance de l'intégration des objets connectés dans notre quotidien est réellement forte. Dans un futur proche, nous serons entourés par un très grand nombre de ces objets. parallèlement il existe peu de systèmes aujourd'hui qui permettent une interaction intel-

ligente avec l'ensemble de ces objets. Il existe peu de moyens simples et directs pour réussir à communiquer avec l'ensemble de ces objets connectés sans devoir se connecter manuellement à eux. De plus tous ces acteurs utilisent un langage différent pour communiquer, il est donc difficile de réussir à centraliser un moyen de communication universel.

SmartDevCom permet de réaliser une intelligence semblable à Jarvis, qui est une intelligence artificielle forte, que l'on peut voir dans les films Iron Man. Cette intelligence peut communiquer avec tout, et peut même deviner ce qu'elle doit faire à partir d'informations qu'elle aura apprise elle-même au cours du temps. Avec le temps, Jarvis est capable d'apprendre et de réaliser des tâches complexes : Les scenarios.

1.3.2 Les scénarios

Les scénarios sont un ensemble d'actions que l'on souhaite faire en simultanée et/ou de manière séquentielle. Par exemple, lorsque l'on souhaite regarder un film, voici un scénario avancé que l'on pourrait réaliser :

— Choix du film
— Choix du volume
— Fermeture des stores
— Extinction de la lumière principale
— Allumage d'une petite lumière tamisée, accordée au thème du film
— Passage du téléphone en mode silencieux

Grâce au système de scénario, il serait possible de réaliser l'intégralité des tâches, seulement en demandant de lancer le film voulu. Nous allons voir comment créer ce genre de scénario.

1.3.3 Analyse et anticipation de requêtes

Afin de créer l'intelligence des objets, l'utilisateur possède deux choix. Il peut créer un scénario manuellement en indiquant les tâches à réaliser parallèlement et séquentiellement. Ou alors il peut laisser l'apprentissage se faire. En effet, Jarvis peut simplement faire une étude statistique des utilisations des objets, et faire une corrélation entre eux. Ainsi il peut voir après plusieurs utilisations, que lorsque l'utilisateur réalise une tâche, il en réalise forcément d'autres. Ceci permet donc de réaliser une intelligence centrale, capable d'être personnalisée automatiquement pour son utilisateur, et de pouvoir réaliser un ensemble de tâches de manière autonome.

Il existe donc un autre type d'objet où est placée cette intelligence, qu'on l'on nommera des centrales. Ces centrales permettent à la fois de pouvoir réaliser ces scénarios, mais c'est aussi grâces à une base de données contenue dans chacune d'elle que l'on pourra communiquer avec le bon objet.

Chapitre 2

Mise en place du protocole de communication

Le premier chapitre de ce rapport a présenté les principales idées que l'on souhaite mettre en place pour avoir une architecture d'objets dynamique et générique. On va maintenant s'intéresser à la manière dont on peut les réaliser.

2.1 Les protocoles existants

Il existe une multitude de supports de communication déjà existants, dont les plus connus sont zigbee, et zwave dans le milieu de la domotique. Il existe une autre alternative qui commence à arriver qui s'appelle AllJoyn qui sera expliqués après. Ces protocoles n'ont pas été utilisés, car ils ne possédaient pas l'ensemble des caractéristiques qui seront décrites plus tard.

2.1.1 Zigbee [9]

ZigBee est un protocole de haut niveau permettant la communication, à basse consommation. Ce protocole est basé sur la norme IEEE 802.15.4 et est utilisée pour les réseaux WPAN (*Wireless Personal Area Networks*). Le principal avantage de ce support est sa capacité à créer un réseau maillé dynamique et autonome, ce qui est très pratique pour créer un réseau d'objet dynamique.

2.1.2 Z-Wave [10]

Z-wave est un protocole qui ressemble beaucoup au zigbee. En effet lui aussi permet de créer un réseau maillé dynamique. Les quelques différences avec le z-wave sont principalement :

— Les bandes de fréquences : 868 MHz, 915 MHz et 2.4 GHz pour le ZigBee ; 868 MHz et 908 MHz pour le Z-Wave
— Le nombre de noeuds dans le réseau : 232 pour Z-Wave et 64000 pour ZigBee.

2.1.3 Alljoyn

Alljoyn est un projet open source international, porté par Qualcomm et présenté pour la première fois en 2011. Ce protocole a pour but de communiquer avec tous les objets autour en s'abstrayant du support de communication, et en étant compatible avec tous les systèmes du marché. Depuis, ce protocole est porté par de nombreuses sociétés telles que Microsoft, LG, Sony, Cisco, etc.

2.2 Le réseau virtuel

Nous allons maintenant nous intéresser aux caractéristiques du protocole qui a été développé dans ce projet. L'un des points clés de la mise en place d'objets pour la domotique est la communication avec ceux-ci. Ces objets doivent en effet être "connectés" pour que l'on puisse les contrôler ou recevoir des messages de leur part. Il est donc nécessaire de mettre en place une architecture réseau qui puisse lier ces objets de façon efficace. Mais la mise en place de ce réseau n'est pas forcément évidente. Si l'on souhaite être indépendant du support de communication, il faut mettre en place des moyens de lier des objets de supports différents. Il est également possible de communiquer avec des objets mobiles. Cela signifie que l'architecture du réseau doit également être dynamique. C'est pourquoi il est indispensable d'avoir un protocole capable de gérer tous ces détails.

2.2.1 Architecture du réseau

Pour bien comprendre comment est structuré un réseau d'objets, il faut d'abord s'intéresser à une architecture simple ne faisant qu'intervenir un seul support de communication. La figure 2.1 représente un réseau créé par des objets connectés possédant une interface Wifi. En utilisant du Wifi *Point d'Accès*, un des objets fait office de serveur (C) sur lequel les autres objets clients viendront se connecter. Avec cette architecture de réseau, l'ensemble des objets peuvent parler entre eux mais leurs messages passera par C. Cette connexion est complètement transparente à l'utilisation puisqu'elle directement gérée par le protocole IP. La communication consistera simplement à créer une *socket* TCP ou UDP entre deux objets. Ils pourront alors parler sans avoir à s'occuper du transport de leurs messages.

Si l'on prend cette fois-ci l'exemple d'une architecture formée par un ensemble d'objets utilisant ZigBee (figure 2.2), on remarque que les objets ne sont pas lié à un unique objet mais à

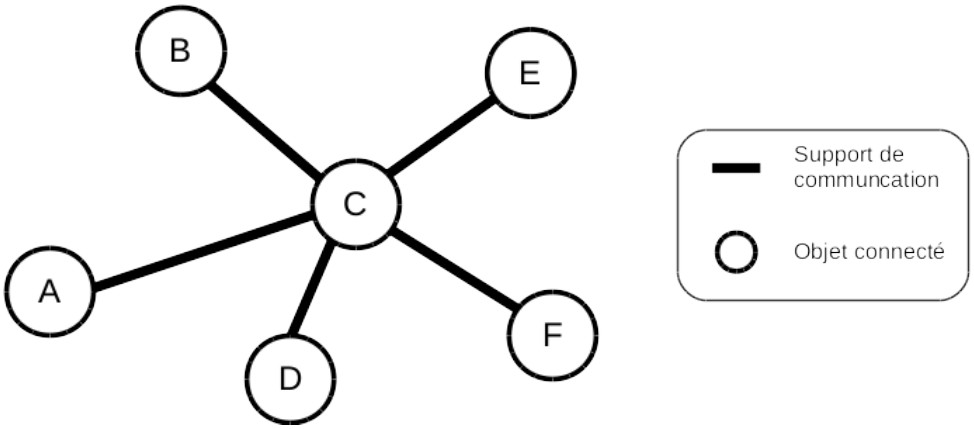


FIGURE 2.1 – Exemple d'architecture réseau d'un point d'accès Wifi (topologie en étoile).

l'ensemble des objets qui sont à sa portée. Un objet peut donc communiquer directement avec un objet voisin mais également avec des objets plus distants s'il existe un chemin dans le graphe du réseau. Encore une fois, toute cette architecture est déjà gérée nativement par les composants ZigBee. Cependant son utilisation est un peu différente d'une utilisation d'objets en Wifi. Certains protocoles de communication utilisant du ZigBee (par exemple Xbee) ne permettent pas d'envoyer des messages à une cible précise. C'est la totalité des objets qui sont présents dans le réseau ZigBee qui recevront le message. Cela signifie qu'il faudra gérer la cible dans le message en lui-même. Cette méthode peut être un inconvénient puisqu'il complexifie la création des messages mais il offre un avantage au niveau du transport de ces messages.

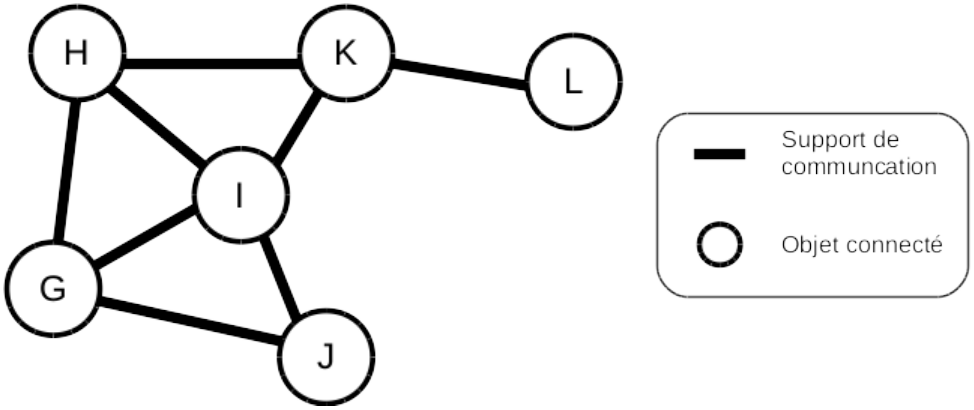


FIGURE 2.2 – Exemple d'architecture réseau d'objet en ZigBee (topologie en maillage).

Ces deux architectures sont les plus courantes dans les technologies de communication sans fils. Certaines de ces technologies offrent même la possibilité de choisir quelle architecture on souhaite utiliser. En domotique, la plupart des produits proposés dans le marché utilise une architecture maillée car elle est la plus adaptée aux tailles des maisons. Mais souvent, ces produits nous limitent à l'utilisation d'une même technologie de communication pour tous les objets de la maison (et également de la même marque). Ceci peut être problématique lorsque l'on souhaite ajouter dans le réseau un objet connecté que l'on aurait fabriqué nous-même. En effet, certains de ces produits utilisent des technologies assez onéreuses. Une façon de résoudre ce problème est

de ne pas limiter le réseau à un unique support de communication mais à mettre en place des outils permettant de connecter entre eux différents supports.

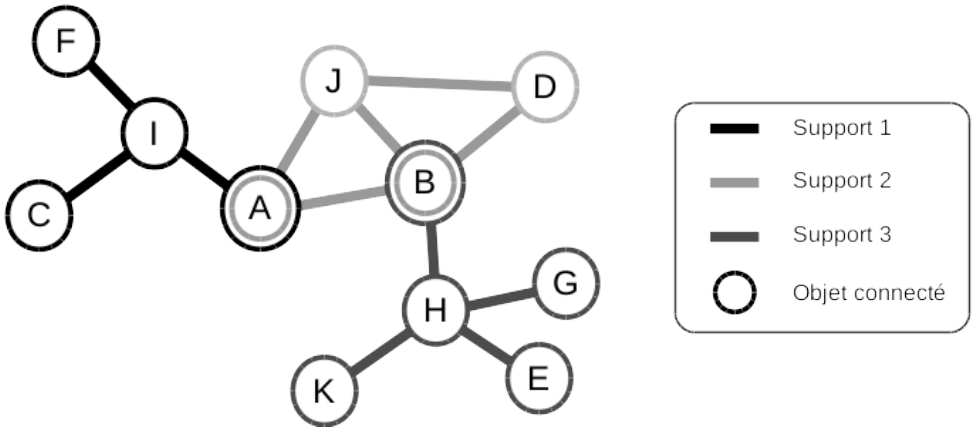


FIGURE 2.3 – Exemple d'architecture réseau mélangeant plusieurs supports de communication.

La figure 2.3 est un exemple de réseau que notre protocole est capable de gérer. On peut voir qu'il permet de lier entre eux 3 supports de communication différents ayant également des architectures de réseau différentes. Le lien entre chacun de ces supports est réalisé par des objets connectés un peu particulier que l'on appelle des *passerelles* (A et B sur le schéma). Ces objets peuvent fonctionner de la même façon que n'importe quel autre objet. Ils ont cependant la possibilité de communiquer sur différents supports. Cela signifie que ces objets sont équipés au minimum de 2 technologies de communications. Par exemple, Wifi + Bluetooth, ZigBee + Ethernet ou même Wifi + Wifi (dans 2 réseaux différents).

2.2.2 Le routage

Même avec une architecture de réseau comme celle de la figure 2.3, il faut laisser la possibilité à chaque objet de pouvoir communiquer avec n'importe quel autre objet. Cela ne sera pas aussi simple qu'avec une architecture ne contenant qu'un seul support.

On va considérer que deux objets connectés sont *voisins* s'ils appartiennent au même support de communication. On souhaite par exemple que l'objet C envoie un message à l'objet F. La première chose à faire est de déterminer qui est la cible. Étant donné que les deux objets appartiennent au même support, il suffit d'utiliser les adresses de support de F et$C$ (l'adresse MAC pour le Bluetooth, l'adresse IP pour le Wifi, ...). Le message sera alors transmit sans aucun problème puisque ce sera la technologie de communication qui s'occupera du transport du message. Maintenant on souhaite que l'objet C envoie un message à l'objet G. Cette fois-ci il va y avoir un problème puisque ces deux objets n'appartiennent pas au même support. L'utilisation des adresses de support ne servira à rien puisqu'elles ne seront pas compatibles. De plus, il est nécessaire de faire intervenir les objets A et B car ils font obligatoirement partie du transport du message.

Ce problème a déjà été abordé lors de la création des premiers réseaux informatiques. Il existe différentes solutions mais la plus connue est sans doute celle qui est actuellement utilisée dans nos ordinateurs : le protocole IP. Cela signifie qu'il faut mettre en place un système similaire à celui qui relie les machines du monde entier. L'ensemble des objets connectés formera donc un *réseau virtuelle* qui s'appliquera comme une surcouche aux technologies de communication existantes. Le nom de ce nouveau protocole des objets connectés est *VIP* (Virtual Internet Protocol) imitant de façon simplifiée le protocole IP. Il est également accompagné du protocole VARP (virtual adress resolution protocol) qui sera détaillé plus tard.

Le protocole VIP

Le protocole VIP définit pour chacun des objets une adresse VIP qui est composée de 3 octets (tableau 2.1). Chacun de ces octets possède une utilité précise. Le premier octet est utilisé pour référencer l'adresse du réseau. C'est une adresse globale qui sert à regrouper ensemble plusieurs sous-réseaux. En pratique, on utilise la même adresse de réseau pour la totalité des objets d'une maison. Cependant, si la maison possède plus de 64516 objets, il faudra au minimum 2 adresses de réseau pour représsenter une maison. Le deuxième octet de l'adresse VIP représente l'adresse de sous-réseau. C'est l'adresse d'un support de communication. Cela signifie que tous les objets d'un même support doivent utiliser la même adresse de sous-réseau. Encore une fois, s'il y a plus de 254 objets dans le même support, il faudra utiliser plusieurs adresses de support. Mais il faudra cependant ajouter des objets passerelles dans le même support pour que tous les objets puissent communiquer entre eux. Le dernier octet de l'adresse VIP va enfin définir l'adresse de l'objet. Cette adresse doit bien sûr être unique pour chaque objet d'un même sous-réseau.

Nom	Taille (bits)	Description
`ADDR_NET`	8	l'adresse de réseau
`ADDR_SUB_NET`	8	l'adresse de sous réseau
`ADDR_DEV`	8	l'adresse de l'objet

Tableau 2.1 – Format d'une adresse VIP

Il existe cependant des valeurs particulières d'adresse VIP :

Les adresses de réseaux Toutes les addresses VIP dont l'adresse d'objet est 0 représente l'adresse du sous-réseau. Cela signifie que l'adresse ne décrit pas un objet mais le sous-réseau lui-même. De la même façon, toutes les adresses VIP dont l'adresse de sous-réseau est 0 représente l'adresse de réseau (l'adresse doit valoir également 0).

Les adresses de broadcast Une adresse de broadcast est une adresse un peu particulière puisqu'elle représente la totalité des objets possédant le même réseau ou sous réseau que l'adresse de broadcast. Elle est définie avec la valeur maximale d'un ou plusieurs octets de son adresse

VIP. Il faut cependant que ces octets soient les derniers de l'adresse.

Les adresses VIP sont donc très similaires aux adresses IP. Elles n'ont cependant pas la même flexibilité puisque IP offre la possibilité de définir des masques de sous-réseaux dynamiques. Il y a également une différence au niveau de la représentation textuelle d'une adresse. Une adresse VIP s'écrit de la façon suivante :

`ADDR_NET:ADDR_SUB_NET:ADDR_DEV`

en remplaçant chaque `ADDR` par sa valeur en hexadécimal. Le tableau 2.2 montre quelques exemples d'adresse VIP.

`0f:34:02`	objet 0x02 du sous-réseau 0x34 du réseau 0x02
`aa:b2:27`	objet 0x27 du sous-réseau 0xb2 du réseau 0xaa
`aa:32:1c`	objet 0x1c du sous-réseau 0x32 du réseau 0xaa
`23:00:00`	adresse du réseau 0x23
`23:4b:00`	adresse du sous-réseau 0x4b du réseau 0x23
`23:4b:ff`	adresse de broadcast du sous-réseau précédent
`23:ff:ff`	broadcast à tous les objets du réseau 0x23
`ff:ff:ff`	broadcast à tout le monde (à utiliser de façon modérée)
`00:00:00`	adresse du réseau virtuel (utile pour le routage)

TABLEAU 2.2 – Exemples d'adresses VIP.

Avec ce système d'adresse pour les objets connectés, il est maintenant possible d'identifier de façon uniforme chaque objet du réseau virtuel mais également de pouvoir gérer la communication entre des objets de supports différents grâce au mécanisme de *routage*. En effet, chaque objet est équipé d'une table que l'on appelle *table de routage*. Ces tables permettent de définir le chemin que va prendre un message entre un objet A et un objet B. Elles contiennent une liste décrivant, pour chaque interface de communication d'un objet, l'adresse vers qui envoyer les données en fonction du destinataire du message. Cela signifie que si l'on souhaite envoyer un message à une certaine adresse, le programme parcourra la table de routage à la recherche de la première règle qui correspondra à l'adresse. Le message sera alors envoyé à la passerelle associée la règle correspondante dans la table. Si aucune règle n'est adapté, il prendra la règle par défaut (adresse `00:00:00`). On peut également mettre des règles dont l'adresse de passerelle est `00:00:00`. Cela signifie que la passerelle est l'objet lui-même (c'est un réseau local à l'objet).

Pour mieux comprendre ce mécanisme, prenons un exemple dans la figure 2.4. L'objet C souhaite envoyer un message à l'objet F. Le tableau 2.3 donne la table de routage de chacun des objets connectés décrits dans la figure. La table qui nous intéresse dans cet exemple est celle de l'objet C car il faut déterminer l'interface de communication qu'il faut utiliser pour émettre le message. Dans le cas de cet objet, la solution est vite trouvée puisqu'il ne possède qu'une seule interface : le support 1. L'adresse de l'objet F est `01:4b:17`. L'objet C va donc chercher l'adresse de réseau correspondante. La première ligne de la table concerne le réseau d'adresse `01:4b:00`. Cette ligne est déjà valide pour l'adresse de l'objet F. Le message doit donc être envoyé à l'adresse de la passerelle de cette ligne. Il se trouve que l'adresse de passerelle est l'adresse par défaut. Cela signifie que l'on peut envoyer le message directement à l'objet F. Il n'y a pas besoin de

passer par des intermédiaires.

Maintenant, on va analyser l'envoi d'un message vers une destination plus lointaine. Prenons le cas de l'envoi d'un message entre l'objet C et l'objet G. L'adresse de l'objet G est `01:cc:42`. Dans la table de routage de C, la ligne concernée va être la deuxième puisque le réseau `01:cc:00` n'est pas connu. Il va donc envoyer le message à l'objet A (la passerelle). Après réception du message par l'objet A, celui-ci va lire sa table de routage pour chercher l'adresse de G. La règle valide pour cette adresse est encore une fois la règle par défaut. Le message va donc être envoyer à l'objet d'adresse `01:e5:33` (l'objet B). C'est ensuite au tour de B de lire sa table de routage. Cette fois-ci le réseau `01:cc:00` est connue. La règle valide dans sa table est donc la deuxième. Le message sera ensuite envoyé à l'objet G.

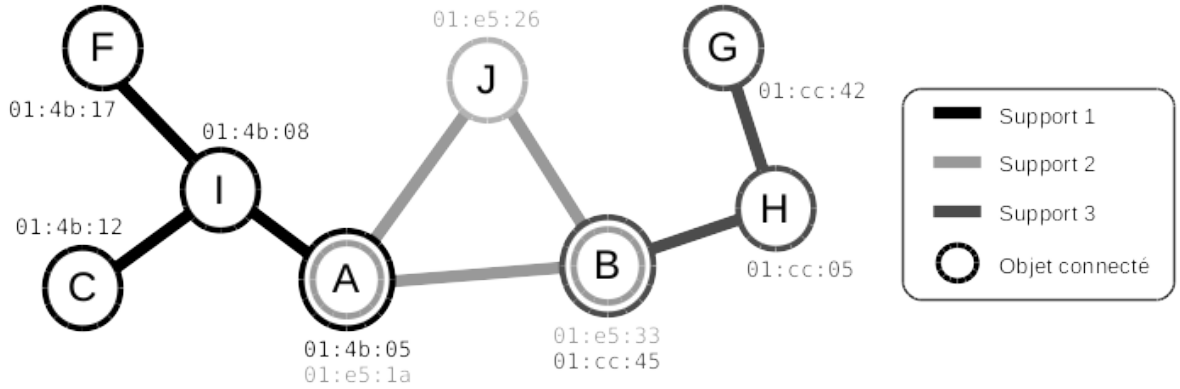


FIGURE 2.4 – Exemple de réseau en spécifiant les adresses VIP.

Objet	Adresse de réseau	Passerelle	interface
A	`01:4b:00`	`00:00:00`	support 1
	`01:e5:00`	`00:00:00`	support 2
	`00:00:00`	`01:e5:33`	support 2
B	`01:e5:00`	`00:00:00`	support 2
	`01:cc:00`	`00:00:00`	support 3
	`00:00:00`	`01:e5:1a`	support 2
C	`01:4b:00`	`00:00:00`	support 1
	`00:00:00`	`01:4b:05`	support 1
F	`01:4b:00`	`00:00:00`	support 1
	`00:00:00`	`01:4b:05`	support 1
G	`01:e5:00`	`00:00:00`	support 3
	`00:00:00`	`01:cc:45`	support 3
H	`01:cc:00`	`00:00:00`	support 3
	`00:00:00`	`01:cc:45`	support 3
I	`01:4b:00`	`00:00:00`	support 1
	`00:00:00`	`01:e5:1a`	support 1
J	`01:e5:00`	`00:00:00`	support 2
	`00:00:00`	`01:e5:1a`	support 2

TABLEAU 2.3 – Les tables de routage des différents objets de la figure 2.4

Comme l'ont montré ces exemples, l'utilisation des tables de routage est un bon moyen pour transporter un message dans la totalité du réseau virtuel. Mais actuellement, elle possède un petit inconvénient. Il est nécessaire d'écrire à la main les tables de routage de chacun des objets faisant passerelle. Il existe cependant des algorithmes de mise à jour dynamique des tables de routage pour automatiser leur création mais ils n'ont pas encore été étudiés dans ce projet.

2.2.3 Un réseau dynamique

Le protocole VIP est suffisant pour mettre en place une architecture d'objets connectés dans une maison mais il manque cependant une étape pour que les messages puissent utiliser correctement les supports de communication. Imaginons que l'on souhaite envoyer un message à un objet appartenant au même support. On connait son adresse VIP. Mais comment fait on pour déterminer à qui appartient l'adresse ? Il est en effet nécessaire de pouvoir faire le lien entre l'adresse de support d'un objet et son adresse VIP. Pour cela, il a été mis en place un protocole nommé VARP (Virtual Address Resolution Protocol) inspiré du protocole ARP. Ce protocole permet d'effectuer des demandes d'adresses VIP ou d'adresses de support. Par exemple, si un objet est en possession d'une adresse de support, il peut récupérer l'adresse VIP correspondante avec une requête VARP. Et à l'inverse, il peut récupérer une adresse de support à partir d'une adresse VIP. Il est également possible de faire une requête multiple en utilisant une adresse de broadcast. Cela signifie que tous les objets acceptant cette adresse répondront à la requête. Cela peut par exemple être utilisé pour mettre à jour une table contenant la liste des couples d'adresses support/VIP des objets voisins. La figure 2.5 et le tableau 2.4 présente quelques exemples d'utilisation du protocole VARP.

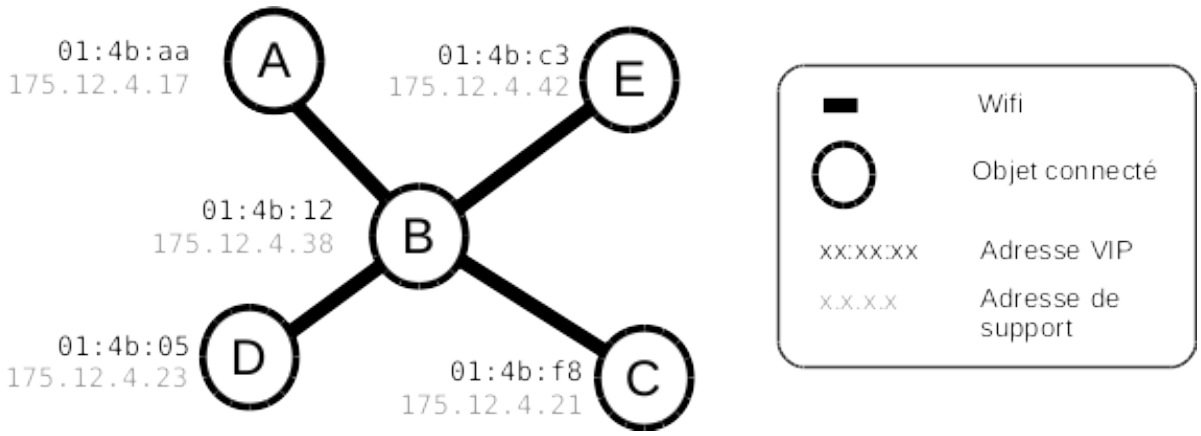


FIGURE 2.5 – Exemple de réseau Wifi d'objets connectés avec leurs adresses IP et VIP.

En faisant des scans réguliers en utilisant le protocole VARP, il est possible de détecter la connexion ou la déconnexion d'objets dans le réseau virtuel. Cela peut être pratique pour rendre le réseau dynamique. Mais lorsqu'un nouvel objet se connecte au réseau, il faut cependant qu'il possède déjà une adresse VIP. De plus, cette adresse doit également respecter l'adresse de sous-réseau du support dans lequel il va se connecter. Sinon, il ne sera pas possible de communiquer

Requête			Réponse		
Objet	Demande	Adresse connue	Objet	Adresse VIP	Adresse IP
A	IP	01 :4b :f8	C	01 :4b :f8	175.12.4.21
A	IP	01 :4b :ff	B	01 :4b :12	175.12.4.38
			C	01 :4b :f8	175.12.4.21
			D	01 :4b :05	175.12.4.23
			E	01 :4b :c3	175.12.4.42
A	VIP	175.12.4.38	B	01 :4b :12	175.12.4.38
A	VIP	255.255.255.255	B	01 :4b :12	175.12.4.38
			C	01 :4b :f8	175.12.4.21
			D	01 :4b :05	175.12.4.23
			E	01 :4b :c3	175.12.4.42

TABLEAU 2.4 – Exemple d'utilisation du protocole VARP par rapport à la figure 2.5

avec ce nouvel objet. La solution à ce problème serait d'affecter automatiquement une adresse VIP à chaque nouvel objet qui rejoint le réseau. Le protocole DHCP (Dynamic Host Configuration Protocol) a été conçu pour ça dans les réseaux Ethernet. Il faut donc créer un protocole similaire pour les objets connectés dans le réseau virtuel. Ce protocole n'a pas encore été étudié et n'est donc pas présent dans l'implémentation actuelle du réseau virtuel.

Avec la mise en place d'un système de connexion dynamique d'objets connectés, il est possible de mettre en place un nouveau type d'objet que l'on appellera *objet mobile*. Ce genre d'objet doit pouvoir se déconnecter d'un sous-réseau lorsqu'il sort de sa portée mais il doit également se connecter automatiquement aux réseaux qui sont dans sa proximité. Ils possèdent donc une adresse VIP variable. En utilisant un identifiant unique pour ces objets, il est possible de les localiser "grossièrement" dans la maison si l'on sait où sont placer les objets fixes. Cette information peut alors être utilisée pour effectuer des actions particulières dans des protocoles de plus haut niveau. Si l'on transforme des montres intelligentes ou encore des smartphones en objet connecté, on peut alors déterminer la localisation d'une personne dans le réseau virtuel.

2.3 La généricité des objets connectés

Avec les protocoles VIP et VARP, nous avons vu comment sont transportés les messages entre n'importe quels objets du réseau virtuel. Il est temps maintenant de passer à un niveau plus haut et de s'intéresser à la composition de ces messages.

2.3.1 Description d'un objet dans le protocole

Jusqu'à présent, tous les éléments du réseau virtuel ont été décrit comme étant des objets connectés. On peut alors se demander pourquoi les différents noeuds du réseau sont qualifiés ainsi. Quels sont leurs caractéristiques ? La définition d'un objet connecté a déjà été abordé

brièvement dans la première partie de ce rapport. Maintenant nous allons voir un peu plus en détails les fonctionnalités de ceux-ci. Tous ces éléments sont définis dans un protocole encapsulé dans le protocole VIP : le protocole SDCP (Smarts Device Communication Protocol). Le tableau 2.5 illustre l'organisation des différentes couches du protocole de SmartDevCom.

<table>
<tr><td rowspan="3">Couche réseau</td><td rowspan="3">Bluetooth</td><td rowspan="3">ZigBee</td><td>Ethernet</td></tr>
<tr><td>IP</td></tr>
<tr><td>UDP</td></tr>
<tr><td rowspan="2">Couche réseau virtuel</td><td rowspan="2">VARP</td><td colspan="2">VIP</td></tr>
<tr><td colspan="2">SDCP</td></tr>
</table>

TABLEAU 2.5 – Représentation des différentes couches réseaux du protocole utilisé par SmartDevCom

Le protocole permet de définir des objets connectés dont on ne connait pas la composition à l'avance. Il est donc nécessaire de définir des règles permettant de décrire n'importe quel objet. Ce protocole doit alors essayer de rester le plus générique possible tout en offrant des fonctionnalités simples d'utilisation. Le protocole définit un objet connecté de la manière suivante :

— il possède zéro, un ou plusieurs capteurs
— il possède zéro, un ou plusieurs actionneurs
— il possède zéro, une ou plusieurs actions
— il définit la façon dont doivent être utilisé chacune de ses actions
— il permet l'exécution de chacune de ses actions

Pour rappel, on considère ici qu'un capteur est un module d'un objet capable de mesurer une grandeur physique de son environnement. Un actionneur est un module capable d'agir sur son environnement. Une action représente une requête ou un ordre que l'on peut donner à un ou plusieurs des modules d'un objet. Étant donné que l'on ne connait pas forcément à l'avance à quoi correspond une action, il est nécessaire de définir précisément les paramètres qu'elle prend ainsi que les informations qu'elle retourne. L'ensemble des ces caractéristiques est récupérable à distance grâce à des requêtes définies dans le protocole SDCP. C'est justement cet ensemble de requêtes qui permet de pouvoir communiquer avec un objet sans avoir à connaitre au préalable à quoi il sert.

2.3.2 Méthode de classification des actions sur les objets

Pour pouvoir utiliser convenablement un objet inconnu, il est cependant nécessaire de pouvoir le ranger dans une certaine catégorie d'objet. En effet, on exécutera une de ses actions uniquement si l'on sait à quoi elle sert. C'est pourquoi le protocole SDCP définit des types de capteurs, des types d'actionneurs et surtout des types d'actions. Dans cette architecture, connaitre le type des capteurs et des actionneurs d'un objet n'est pas quelque chose d'indispensable. Lors de l'utilisation d'un objet, ce qui nous importe c'est l'exécution des actions. Si l'on prend l'exemple d'un objet capable d'allumer la lumière en appuyant sur un interrupteur mural. La demande de liste des actionneurs va retourner le servomoteur qui s'occupe de changer l'état du bouton. La

demande de liste d'actions va retourner l'action permettant d'allumer ou d'éteindre la lumière. On remarque que le type de l'actionneur n'a que peu d'intérêt par rapport au type de l'action car ce dernier apporte une information bien plus précise.

Mais le problème le plus important dans ce protocole est la définition des types des actions. Ce type doit valider deux contraintes : il doit définir précisément le rôle de l'action et il doit également être générique. Cela signifie par exemple que si un nouvel objet apparait dans le réseau avec un type d'action inconnu, les autres objets doivent pouvoir comprendre de la façon la plus précise possible à quoi sert cette action. Cela pose demande donc beaucoup de réflexion sur la méthode à utiliser pour construire les types d'actions.

Une solution intéressante pour résoudre ce problème serait de construire un arbre de types d'action. Cela permettrait de pouvoir définir des actions de façon très précise (lorsque l'on s'approche des feuilles de l'arbre) et également conserver un certain ordre d'idée pour des types qui n'existent pas encore dans l'arbre (on connaitrait au moins l'une des branches de l'arbre). Mais la construction de cet arbre de type n'est, encore une fois, pas évidente. Pour cette construction, une première approche a été faite en utilisant des outils de statistique. Il existe en effet beaucoup d'algorithmes de clusterisation qui crée des structures sous forme d'arbres. Le problème de ces algorithmes est qu'ils utilisent des données quantitatives. Dans notre cas, les données sont purement qualitatives. Pour ce genre de données, une des méthodes consiste à lister un ensemble d'actions le plus varié possible et de rechercher des critères pouvant discriminer ces actions. Cette méthode est intéressante mais la recherche des critères discriminants est presque aussi compliqué que d'essayer de construire l'arbre des types à la main.

Finalement, l'arbre des types d'actions a été écrit manuellement en essayant de faire le tour des types d'actions que l'on peut potentiellement rencontrer avec des objets connectés. Voici un extrait de la liste des types d'actions actuellement gérées (le vrai arbre est sur 4 niveaux) :

- ambiance
  - action kinestésique
  - action visuelle
  - action auditive
- sécurité
- multimédia
  - audio
  - vidéo
- électroménager
  - alimentaire
- accessibilité (humain)
  - déplacement
  - ouverture de porte
  - localisation
- entretien

2.3.3 Gestion dynamique d'objets

Cette méthode de gestion des types d'actions est déjà suffisante pour la plupart des objets que l'on serait susceptible de créer. Mais il est cependant possible d'aller encore plus loin en créant des objets spéciaux agissant comme des bases de données. Ces objets auront pour tâche de mettre à jour un arbre de type regroupant tous les types que l'on peut rencontrer dans le réseau virtuel. Cette base sera bien évidemment dynamique pour que des nouveaux objets puissent ajouter de nouveau type. Ces objets peuvent également rassembler d'autres informations comme par exemple la liste des actions de tous les objets. Ce regroupement d'informations peut être très utile pour d'autres objets dont l'unique but est de contrôler des objets existants.

Le protocole SmartDevCom réunit assez d'outils pour pouvoir créer des objets connectés très variés. Il permet d'avoir une structure solide sur laquelle on peut alors mettre en place tout une hiérarchie d'objets connectés allant du simple objet appuyant sur un bouton jusqu'à des objets intégrant de l'intelligence artificielle et dédié au contrôle des autres objets. Il n'y a alors plus aucune limite dans la dynamicité et la réactivité du réseau virtuel.

Chapitre 3

La création des objets

3.1 Reconnaissance vocale : Application Android

3.1.1 Etat de l'art

Tout projet implique une phase de recherche, afin d'effectuer un état de l'art sur ce qu'il a déjà été fait auparavant dans ce domaine. Afin de rechercher ce qu'il fallait pour cet état de l'art, il a donc fallu décomposer en plusieurs parties ce que devait réellement réaliser cette reconnaissance. Ces parties étaient :

— Activation de la reconnaissance vocale automatiquement
— Gestion de la chaîne de caractères comprise par la reconnaissance

De base sur Android, il est possible d'activer la reconnaissance vocale automatiquement grâce aux mots clefs «OK Google». Souhaitant appeler notre intelligence domotique Jarvis, il était peu souhaitable de dire ces mots afin d'activer la reconnaissance. Après recherches, il s'avère qu'il n'existe aucun moyen simple de changer ces mots clefs, puisque les seuls moyens sontsoit d'avoir un téléphone rooté, autrement dit où l'on a tous les accès afin de changer le bon fichier de configuration. Soit d'avoir un téléphone de la marque Motorola, qui intègre le changement de ces mots directement dans son firmware. C'est pourquoi pour activer la reconnaissance vocale, nous sommes obligés de prononcer les mots «OK Google».

3.1.2 Activation de la reconnaissance vocale

Afin de pouvoir démarrer et utiliser la reconnaissance vocale, il faut un ensemble d'éléments :
— Autovoice
— Tasker

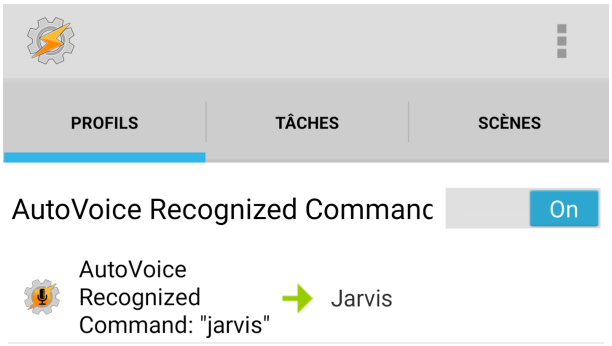


FIGURE 3.1 – Autovoice + Tasker

Autovoice est une application Android qui permet d'utiliser la reconnaissance vocale de Google, et de se servir de la chaîne de caractères reconnues pour en faire ce que l'on souhaite. Par ailleurs, on peut la paramétrer de telle sorte qu'elle réagisse seulement si une suite de mots clefs est reconnue, comme sur la figure 3.1. Afin de pouvoir communiquer avec Jarvis, le seul mot clef choisi a donc été son nom. Du côté de Tasker, c'est une application d'automatisation très connue sur Android. Elle permet de réaliser des tâches en réponse à un événement donné. Ainsi il est possible d'utiliser comme événement Autovoice, et comme tâche, une application qui permet de traiter la chaîne de caractères : SpeechToText, comme on peut le voir sur la figure 3.2.

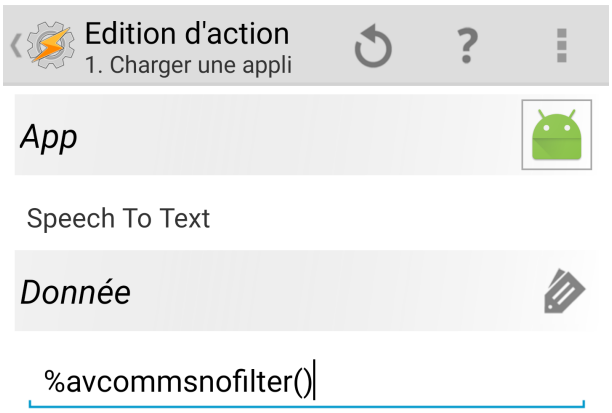


FIGURE 3.2 – Tâche jarvis

3.1.3 SpeechToText

La base de données

SpeechToText est une application Android qui permet donc d'interpréter une chaîne de caractères, et d'envoyer une trame suivant le protocole SDCP au bon objet connecté. L'application possède deux bases de données dynamiques. La première possède l'ensemble des objets intelligents. Cette base de données est stockée dans un fichier JSON.

```
{
    "smartDevices" : [
      {
        "device": {
          "idAction": "0x111",
          "bleName": "sdc_Lisa"
        }
      }]
}
```

FIGURE 3.3 – Base de données des objets intelligents

Cette base de données comprend une liste des objets connectés proches de nous. Comme on peut le voir un objet est caractérisé par deux choses : l'identifiant de l'action qu'il peut réaliser, ainsi que le nom de l'objet bluetooth auquel il faut se connecter pour réaliser l'action.

```
{
    "voiceCommands" : [
      {
        "command" : {
            "voiceActivation" : "Quelle est la temperature",
            "voiceDiction" : "la temperature est",
            "idAction" : "0x111"
        }
      }]
}
```

FIGURE 3.4 – Base de données des commandes vocales

Ainsi une commande vocale est caractérisée par une chaîne de caractères à reconnaître, une chaîne de caractères à dire par synthèse vocale, afin de donner un feedback sur ce qu'il se passe à l'utilisateur. Elle contient elle aussi l'identifiant de l'action à réaliser, afin de faire le lien avec l'autre base de données.

Dès que la phrase est analysée par la reconnaissance vocale, il est nécessaire de reconnaître la requête que l'utilisateur souhaite réaliser. Pour ce faire, l'application réalise un algorithme sur la chaîne reçue, dont le but est d'évaluer la similarité avec l'ensemble des phrases possibles. La distance minimale caractérisera la requête faite par l'utilisateur. Il existe différents algorithmes, tel que : Levenshtein, ou JaroWinkler.

La distance de Levenshtein

C'est le plus simple des deux algorithmes. Cette distance calcule le nombre de différences qu'il y a entre deux chaînes de caractères. Ces différences peuvent être le remplacement, la suppression,

ou l'insertion d'un caractère.

La distance de JaroWinkler

Celle-ci part de la distance de Jaro dont l'équation est [1] :

$$d = \frac{1}{3}(\frac{m}{|s1|} + \frac{m}{|s2|} + \frac{m-t}{m}) \tag{3.1}$$

où :
— $|s_i|$ est la longueur de la chaîne de caractères de la chaîne 'i'
— m est le nombre de caractères *correspondants* dans les 2 chaînes
— t est le nombre de *transpositions* nécessaires de ces caractères partagés

Deux caractères identiques de s_1 et s_2 sont dit *correspondants* lorsque leur éloignement dans leur chaîne respective ne dépasse pas :

$$\lfloor\frac{max(|s_1|,|s_2|)}{2}\rfloor - 1 \tag{3.2}$$

Le nombre de *transpositions* est obtenu en comparant le i-ème caractère *correspondant* de s_1 avec le i-ème caractère *correspondant* de s_2. Le nombre de fois où ces caractères sont différents, divisé par deux, donne le nombre de transpositions.

On calcule ensuite la distance avec cette équation :

$$d_w = d_j + (l_p(1 - d_j)) \tag{3.3}$$

avec :
— d_j, la distance de Jaro entre s_1 et s_2
— l, la longueur du préfixe commun (avec un maximum de 4 caractères)
— p, un coefficient qui permet de favoriser les chaînes avec un préfixe commun. Winkler propose pour valeur $p = 0.1$.

Wikipédia donne un très bon exemple pour comprendre cette distance. Ils appliquent cette algorithme pour calculer la distance entre MARTHA et MARHTA. Pour cela ils dressent un tableau :

On a :
— $m = 6$ (Nombre de 1 dans la matrice)
— $|s_1| = 6$
— $|s_2| = 6$
— Les caractères correspondants sont {M,A,R,T,H,A} pour s_1 et {M,A,R,H,T,A} pour s_2.

On a donc 2 couples (T/H et H/T) de caractères *correspondants* différents, soit deux demi-

	M	A	R	T	H	A
M	1	0	0	0	0	0
A	0	1	0	0	0	0
R	0	0	1	0	0	0
H	0	0	0	0	**1**	0
T	0	0	0	**1**	0	0
A	0	0	0	0	0	1

FIGURE 3.5 – Exemple pour la distance de JaroWinkler

transpositions. D'où $t = \frac{2}{2} = 1$.

La distance de Jaro est :

$$d_j = \frac{1}{3}\left(\frac{6}{6} + \frac{6}{6} + \frac{6-1}{6}\right) = 0.944 \tag{3.4}$$

La distance de Jaro-Winkler avec $p = 0.1$ avec un préfixe de longueur $\ell = 3$ devient

$$d_w = 0.944 + (3 * 0.1(1 - 0.944)) = 0.961 \tag{3.5}$$

Etude de l'algorithme le plus approprié

La meilleure distance pour notre projet a été trouvée en faisant une série de tests, en utilisant à chaque fois les deux algorithmes. Le test consiste à prendre plusieurs phrases présentes dans le fichier des commandes vocales, et à les dire un grand nombres de fois, avec différents tons de voix, différentes articulations, et différents volumes. Ensuite, pour chacune de ces requêtes, une liste des différentes phrases comprises par la reconnaissance vocale est établie. Les deux algorithmes sont alors appliqués, pour voir lequel est le plus robuste, et nous donne la correspondance la plus grande entre la requête souhaitée, et les commandes comprises. Il s'avère que l'algorithme le plus simple était le plus robuste pour notre étude, c'est donc la distance de Levenshtein qui a été gardée.

Maintenant que nous sommes capables de savoir quelle requête l'utilisateur souhaite envoyer, et à quel objet, il faut savoir comment l'envoyer.

3.1.4 Intégration du protocole avec JNI

Le protocole a été construit pour être sur systèmes embarqués, avec le langage C++. Or l'objet reconnaissance vocale doit lui aussi l'utiliser afin de pouvoir communiquer convenablement avec les autres objets. Ainsi deux choix étaient possibles :

— Réécrire le protocole sous Android

— Intégrer le protocole en réalisant une interface pour Android

C'est la deuxième option qui a été choisie, celant évitant de refaire une grosse partie du travail déjà réalisée.

Il faut savoir que le système Android fonctionne grâce à une machine virtuelle Java : *Art* (Pour Android Run Time). Cette machine n'exécute de base à priori que du byte code Java, et non de l'assembleur que l'on pourrait obtenir en compilant un langage natif comme le C++. Grâce à *JNI*, pour Java Native Interface, il est possible d'interfacer du code java, avec du code natif. Pour cela il suffit de compiler le code natif pour en faire une bibliothèque, qu'il suffira se charger dans le programme. Cette bibliothèque contient un ensemble de fonctionnalités déjà compilé, et qui peut donc être utilisé directement dans le programme.

Intégration de la bibliothèque [2]

La première chose à faire est de créer une classe dans le code Java, elle contiendra plusieurs directives importantes, comme le chargement de la bibliothèque, mais aussi le prototype de la fonction *fromCpp* qui sera défini dans le code natif, cela permettra de créer le lien entre les deux langages. Il faut ensuite demander au compilateur java de créer un fichier d'en-tête à partir de cette classe, qui sera utilisée dans notre bibliothèque plus tard.

Il faut ensuite écrire le corps de la fonction *fromCpp*, dans un autre fichier C++. Afin de faire le lien avec le code java, il faut inclure le fichier d'en-tête tout juste créé. Par la suite, nous devons compiler ce code en précisant que nous souhaitons obtenir une bibliothèque et pas un fichier exécutable. Cela nous donnera un fichier *.so* sous Linux, ou un fichier *.dll* sous Windows. Ce fichier devra être placé à la racine du projet, afin que la machine virtuelle java puisse le trouver pendant l'exécution.

Dès lors il sera possible d'utiliser du code natif, et donc de pouvoir se servir du protocole afin de pouvoir communiquer avec les différents objets, et cela grâce au bluetooth d'Android.

3.1.5 Communiquer avec le protocole BLE

Fonctionnement du BLE

L'application Android a donc pour but d'utiliser le protocole Bluetooth Low Energy (BLE), pour transmettre les paquets aux différents objets connectés. Afin de pouvoir se servir de ce bluetooth, il faut commencer par effectuer un scan pour lister tous les objets supportant ce protocole aux alentours. Une fonction de callback est appelée pour chaque appareil détectée. On peut ensuite se connecter à cet appareil, afin d'être appairé avec celui-ci.

Tous les appareils utilisant le protocole BLE utilisent le GATT (*Generic Attribute Profile*), afin d'envoyer et recevoir des données sur le réseau. Il faut donc obtenir cette interface pour

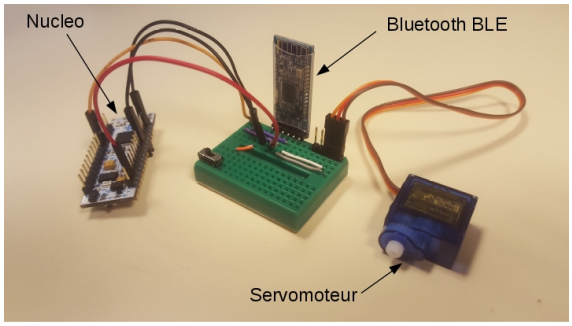


FIGURE 3.6 – Actionneur intelligent

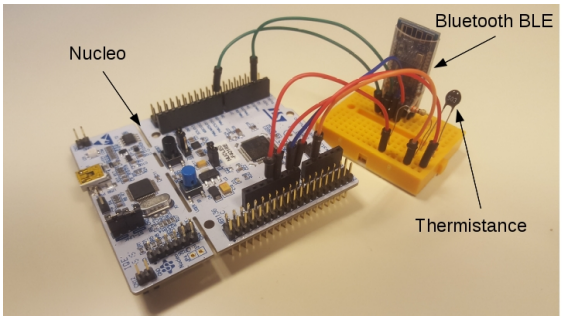


FIGURE 3.7 – Capteur intelligent

pouvoir communiquer en BLE. Une fois l'interface GATT obtenue, il est possible de récupérer l'ensemble des caractéristiques. Les caractéristiques sont l'ensemble des valeurs de l'appareil que l'on peut lire ou écrire. Elles sont composées d'un UUID (*Universally Unique Identifier*), d'une propriété qui permet d'avoir une description lisible de la caractéristique, et enfin d'une valeur. Par conséquent, si l'on souhaite obtenir la valeur de température d'un objet, il faut récupérer la caractéristique dont la propriété est "température", puis simplement récupérer sa valeur.

Cette procédure ne peut être simplifiée, il est donc obligatoire d'effectuer un scan pour voir l'ensemble des objets bluetooth dans les environs, même si nous les connaissons déjà. Il est donc impossible via le BLE d'android de se connecter directement à un appareil à partir de son adresse MAC bluetooth, ce qui est dommage, car nous devons obligatoirement réaliser un scan, afin de communiquer avec un objet proche de nous, même si nous le connaissons.

3.2 Les objets intelligents

Maintenant que les différentes briques ont été posées, il est possible de créer les différents objets intelligents.

Il est possible de voir sur les figures 3.6 et 3.7 que les deux objets ne possèdent pas la même carte de développement Nucleo. La carte présente pour l'actionneur est une F303K8, tandis que celle du capteur est une F401RE. De manière générale, la première possède moins de composants, mais est plus petite, ce qui est largement suffisant pour la création de nos objets connectés.

L'actionneur présent sur la figure 3.6 est composé d'un servomoteur. Cet objet a été utilisé en tant qu'interrupteur intelligent. Le capteur présent sur la figure 3.7 est quant à lui composé d'une thermistance, afin de pouvoir récupérer la température de la pièce.

Perspectives

Beaucoup de choses ont été faites dans ce projet, mais il reste encore plus de choses à faire avant de pouvoir obtenir un vrai réseau d'objets pour la domotique.

Dynamicité du protocole

Avec l'ensemble des outils qui a été défini par les protocoles VIP, VARP, VDHCP et SDCP. Il est théoriquement possible de créer tous les objets que l'on peut imaginer. Mais le protocole n'est pas encore complètement implémenté. Il manque encore la gestion dynamique des tables de routage, la création du protocole VDHCP (pour l'affectation automatique d'adresse VIP) et la gestion des interfaces Wifi. Si l'on souhaite pouvoir profiter pleinement des fonctionnalités du protocole, il est donc indispensable de l'implémenter complètement pour bénéficier d'un réseau virtuel dynamique.

Création d'un réseau virtuel

La plus grande partie du projet a été consacrée à la réalisation du protocole. Mais la partie la plus intéressante est cependant lorsque l'on commence à créer un ensemble d'objets et de les mettre en réseau. On atteint alors une étape beaucoup plus visuelle et donc beaucoup agréable pour travailler. Malheureusement il était indispensable de mettre d'abord en place les briques de base du protocole avant de pouvoir commencer un objet. Mais une fois ces problèmes réglés, il sera alors possible d'équiper entièrement un ensemble de maisons en domotique, afin de créer ce réseau virtuel.

Ajout d'une intelligence

Une fois qu'a été mis en place un réseau virtuel contenant un ensemble suffisant d'objets connectés, on a alors la possiblité d'obtenir des informations sur la maison ou d'effectuer certaines actions. Mais à ce niveau là, on peut alors monter d'un cran l'intelligence de la maison en créant

des objets capables d'analyser les données des autres objets et de pouvoir effectuer des actions en conséquence. Cela signifie que l'on peut mettre en place une sorte d'intelligence artificielle dans certains objets. La maison serait alors capable d'anticiper certaines des requêtes des utilisateurs.

Création d'un logiciel intégré

Ce logiciel permettra d'automatiser la création de tous les objets connectés possibles. Il contiendra une base de données de tous les objets que l'on peut créer, avec la liste de leurs différents capteurs et actionneurs. Cette base de données contiendra la références de ces modules afin de pouvoir les commander sur Internet sur des sites de références. Enfin, il permettra d'envoyer le bon programme sur l'objet automatiquement.

Conclusion

Le but de ce projet était de mettre en place un réseau d'objets connectés équipés de capteurs et d'actionneurs dédiés à la domotique. Mais pour pouvoir réaliser ce genre de réseau, il est indispensable de mettre en place un protocole de communication pour que tous les objets se comprennent. Étant donné que souhaitions avoir aucune contrainte lors de la création des objets connectés, il a fallu écrire un protocole qui soit le plus générique et dynamique possible. C'est pourquoi la tâche la plus importante du projet a été la mise en place du protocole. Actuellement, celui-ci est opérationnel bien qu'il n'intègre pas encore toute la dynamique que nous souhaitions.

Ce projet a donc nécessité une grande partie de réflexion et d'analyse pour concevoir un protocole générique. Il a requis beaucoup de connaissances dans le domaine du réseau mais également un peu dans le domaine des statistiques pour l'organisation des types d'actions. Mais le projet n'a pas été uniquement théorique. L'implémentation du protocole et la création d'objets ont fait intervenir des technologies assez récente comme la norme C++14 pour le protocole, la cross-compilation avec Docker pour programmer sur ARM, JNI pour intégrer du code natif sous Android, des outils de programmation temps réel ainsi que des bibliothèques d'Android et d'Mbed. Ce projet a donc été très enrichissant d'une part, pour les nombreux domaines de l'informatique embarqué qu'il a fait intervenir et d'autre part, pour la fusion et la concrétisation de beaucoup de connaissances que l'on a acquise à l'ISIMA. Cet enrichissement ne va cependant pas s'arrêter là puisque l'étape suivante est maintenant la création de vrais objets qui utiliseront le travail qui a été réalisé dans ce projet.

Lexique

actionneur — Dans le cadre de ce rapport, un actionneur désigne un module d'un objet connecté capable d'agir sur son environnement.

Address Resolution Protocol (ARP) — Le protocole ARP est un des protocoles de la famille IP. Il permet de faire le lien entre une adresse IP et une adresse MAC.

Bluetooth — Bluetooth est un standard de communication permettant l'échange de données sur une très courte distance (de l'ordre de 10 mètres).

Bluetooth Low Energy (BLE) — Le BLE est une amélioration du Bluetooth visant à réduire considérablement la consomation tout en conservant les mêmes caractéristiques que le bluetooth classique.

capteur — Dans le cadre de ce rapport, un capteur désigne un module d'un objet connecté capable de mesure une grandeur physique de son environnement.

Ethernet — Ethernet est un protocole de communication qui permet de mettre en place des réseaux locaux. Le transfert de donnée se fait alors par l'envoi de paquets

Internet Protocol (IP) — IP est une famille de protocole dédié à la création d'un système d'adressage unique pour la mise en place de réseau d'ordinateurs.

support de communication — Dans le cadre de ce rapport, un support de communication désigne une des technologies de communication existante (ex : Wifi, Bluetooth, ZigBee, Ethernet, ...).

UDP UDP (user datagram protocol) est un protocole réseau utilisé pour transférer des informations d'une application à une autre. Les données transférées ne sont cependant pas contrôlées.

voisin Un objet voisin est un objet connecté appartenant au même support de communication que l'objet comparé.

Wifi Le Wifi est un ensemble de protocole de communication sans fil permettant la communication entre périphériques à distance moyenne (de l'ordre d'une centaine de mètres). En fonction du protocole, le débit de la communication peut être relativement importante.

ZigBee ZigBee est un protocole de haut niveau permettant la communication avec de petite radios. Sa consomation est faible, sa portée est d'environ 10 mètres. Il est principalement utilisé dans la domotique.

Bibliographie

[1] Algorithme de la distance de Jaro-Winkler
Disponible sur http ://www.wikiwand.com/fr/Distance$_{d}e_{J}aro - Winkler$

[2] Comment intégrer du code C++ dans du java
Disponible sur https ://www3.ntu.edu.sg/home/ehchua/programming/java/JavaNativeInterface.html JNI

[3] Texas Instruments, 2013, *Dasheet of CC2541.*
Disponible sur http ://www.ti.com/lit/ds/symlink/cc2541.pdf

[4] Blutetooth™, 2003, *Bluetooth Network Encapsulation Protocol specification.*
Disponible sur https ://www.bluetooth.org/docman/handlers/DownloadDoc.ashx ?doc$_{i}d = 6552$

[5] Bolutek, 2014, *CC41-A Specification.*
Disponible sur http ://img.banggood.com/file/products/20150104013145BLE-CC41-A%20Spefication.pdf

[6] Bolutek, 2014, *CC41-A AT Commands.*
Disponible sur https ://halckemy.s3.amazonaws.com/uploads/document/file/94325/FE85NGOIH90OKGT.pdf

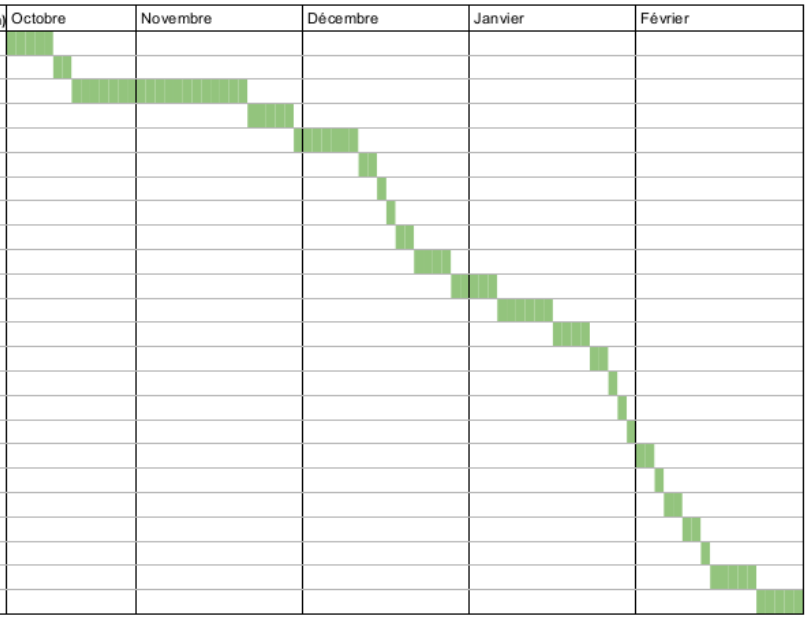
[9] Spécifications du support et du protocole zigbee
Disponible sur ://www.wikiwand.com/fr/ZigBee

[10] différence entre le zigbee et zwave
Disponible sur http ://www.domagency.fr/index.php/infos/faqs-domotique/item/116-quelles-sont-les-diff%C3%A9rences-entre-le- zigbee-et-le-z-wave ?

Diagrammes de Gantt

Prévisionnel

Tâches	Dépend.	Durée (h)
0 - Documentation		20
1 - Ecriture du protocole		8
2 - Implémentation du protocole sous linux	1	80
3 - Implémentation du protocole sous android	1, 2	20
4 - Implémentation du protocole sous arduino	1, 2	30
5 - Prise en main des ESP8266		6
6 - Prise en main des HM-10		4
7 - Cablage d'une carte relai		4
8 - Création d'un objet connecté relai	4, 7	6
9 - Création d'un objet passerelle Raspberry	2	14
10 - Création d'un objet passerelle serveur	2	20
11 - Création d'un objet android reconnaissance vocale	3	24
12 - Interface de commande web	10	20
13 - Réalisation de pièce pour interrupteur commandé		6
14 - Montage de l'objet interrupteur commandé	13	4
15 - Programmation de l'objet interrupteur commandé	4, 5, 6	4
16 - Hack d'une lampe RGB pour créer un objet connecté		4
17 - Programmation de l'objet lampe RGB	4, 5, 6	6
18 - Montage objet télécommande infrarouge		4
19 - Programmation objet télécommande infrarouge	4, 5, 6	8
20 - Prise en main du GSM		8
21 - Montage d'un GPS connecté		4
22 - Programmation d'un GPS connecté	20	20
23 - Coque ZigBee		20
	total :	344



Réel

Tâches	Dépend.
0 - Documentation	
1 - Ecriture du protocole	0
2 - Implémentation du protocole sous linux/nucleo	1
3 - Implémentation du protocole sous android	1, 2
4 - Prise en main des HM-10	
5 - Création d'un objet android reconnaissance vocale	
6 - Réalisation de pièce pour interrupteur commandé	
7 - Montage de l'objet interrupteur commandé	6
8 - Programmation de l'objet interrupteur commandé	2, 4, 6
9 - Hack d'une lampe RGB pour créer un objet connecté	
10 - Programmation de l'objet lampe RGB	2, 4, 9