الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي والبحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

N° Réf :.....

Centre Universitaire

ABD ELHAFID BOUSSOUF MILA

Institut des Sciences et Technologie Département de Mathématiques et Informatique

Mémoire préparé en vue de l'obtention du diplôme de Master

En : Informatique

Spécialité : Sciences et Technologies de l'Information et de la Communication (STIC)

Développement d'un système de stationnement intelligent basé agents

Préparé par : Abderezzek Nora

Soutenue devant le jury

Président : Boukhechem Nadhir	**M.C.B**	**C.U.Abd Elhafid Boussouf**
Examinateur : Nardjes Bouchemal	**M.C.A**	**C.U.Abd Elhafid Boussouf**
Encadreur : Djaaboub Salim	**M.C.B**	**C.U.Abd Elhafid Boussouf**

Année Universitaire : 2021/2022

Remerciement

Avant tout. Nous remercions **ALLAH** *pour son aide, ses innombrables dons et pour nous avoir donné la force, le courage et la volonté d'accomplir ce travail.*

Mes remerciements vont tout d'abord à mon encadreur **Mr "Dr. DJAABOUB SALIM"** *qui a accepté de m'encadrer, ainsi que pour son soutien, ses encouragements, ses orientations et ses précieux conseils tout au long de la réalisation de ce Mémoire.*

Je tiens à exprimer ma gratitude aux membres du jury pour leur patience, leur disponibilité et surtout pour leur acceptation d'examiner et d'enrichir ce travail.

Enfin, je tiens à remercier tous ceux qui m'ont aidé de près ou de loin, merci beaucoup.

Merci

Dédicaces

Je dédie ce travail à :

En particulier mon cher mari Khaled, qui m'a toujours soutenue, encouragé et aidé, que Dieu le protège.

A mes chers enfants ABDERRAHIM ET YAHIA qui sont mon espoir dans cette vie, que Dieu les protège.

A ma chère mère, que Dieu la préserve, et à l'âme de mon cher père, que Dieu ait pitié de lui, qui m'a soutenu depuis l'enfance.

A mon frère et sa famille et Mes sœurs et leurs familles pour leurs soutiens et encouragements.

A tous ceux qui aiment Nora et lui souhaitent bonne chance et succès.

Nora

Résumé

Les derniers développements dans le domaine de la technologie et de la communication, en particulier l'émergence de nouveaux concepts tels que l'IoT (Interne of Things), ont conduit à un grand changement dans tous les domaines de la vie telle que la médecine, le commerce, l'agriculture, les transports, la gestion des villes, etc. Le stationnement intelligent (Smart Parking) est l'un des exemples dans lequel ces nouvelles technologies prouvent être utilisées pour simplifier la tâche de recherche et de réservation de places de stationnement dans les grandes villes et pour donner une meilleure gestion des parkings. Dans ce travail nous avons développé un système de stationnement intelligent basé sur les technologies de l'IoT et de Système Multi-Agents (SMA). Le système réalisé est composé de deux sous-systèmes. Le premier est un système basé IoT qui joue le rôle d'un gestionnaire de parking. Ce système peut être installer dans les parkings de la ville. Le deuxième est un système basé agents qui a pour le rôle de la recherche et la réservation de places de parking dans les différents parkings connectés. Notre système va permettre aux conducteurs à travers une application mobile et avec quelques clics de rechercher et de réserver une place de parking. En outre il va permettre aux gestionnaires d'augtmenter la rentabilité de leurs parkings. Pour la réalisation de notre système différents outils des deux technologies, IoT et SMA, ont été utilisés.

Mots clés : Systèmes de stationnement intelligent, Internet des objets, IoT, Systèmes Multi-Agents (SMA), Arduino, Jade.

Abstract

The last developments in the field of technology and communication, in particular the emergence of new concepts such as IoT (Internal of Things), have led to great change in all areas of life such as medicine, commerce, agriculture, transportation, city management, etc. Smart Parking is one of the examples in which these new technologies are proving to be used to simplify the task of finding and booking parking spaces in large cities and to give better management of parking. In this work we have developed an intelligent parking system based on IoT and Multi-Agent System (MAS) technologies. The realized system is composed of two subsystems. The first is an IoT-based system that plays the role of a parking manager. This system can be installed in any parking of the city. The second is an agent-based system whose role is to search and reserve parking spaces. Our system will allow drivers through a mobile application and with a few clicks to search and reserve a parking space. In addition, it will allow managers to increase the profitability of their parkings. To realize our system, different tools of the two technologies, IoT and SMA, were used.

Keywords: Smart Parking Systems, Internet of Things, IoT, Multi-Agent Systems (MAS), Arduino, Jade.

ملخص

أدت التطورات الأخيرة في مجال التكنولوجيا والاتصالات، ولا سيما ظهور مفاهيم جديدة مثل أنترنت الأشياء (IoT)، إلى تغيير كبير في جميع مجالات الحياة مثل الطب، التجارة، الزراعة، النقل، تسيير المدن، إلخ. مواقف السيارات الذكية هو أحد الأمثلة التي أثبتت فيها هذه التقنيات الجديدة فعاليتها في تسهيل مهمة البحث عن أماكن لوقوف السيارات وحجزها في المدن الكبيرة وكذا السماح بتسيير أفضل لهذه المواقف. في هذا العمل قمنا بتطوير نظام ذكي لركن السيارات يعتمد على تقنيات إنترنت الأشياء والأنظمة المتعددة الوكلاء. يتكون النظام المنجز من نظامين فرعيين. الأول هو نظام قائم على إنترنت الأشياء ويلعب دور مسير موقف السيارات. يتم تركيب هذا النظام في مختلف مواقف السيارات عبرالمدينة. النظام الثاني هو نظام قائم على الوكيل ويمثل دوره في البحث وحجز أماكن وقوف السيارات. سيسمح نظامنا للسائقين من خلال تطبيق للهاتف المحمول وببضع نقرات بالبحث عن مكان لوقوف السيارات وحجزه. كما سيسمح لمسيري المواقف بزيادة ربحية مواقف السيارات الخاصة بهم. لإنجاز نظامنا، تم استخدام تقنيات وأدوات مختلفة من تكنولوجيات الـ IoT والأنظمة المتعددة الوكلاء.

الكلمات المفتاحية : انترنت الأشياء , الأنظمة متعددة الوكلاء ,أنظمة وقوف السيارات الذكية, أردوينو,Jade .

Table des matières

Liste des figures

Liste des tableaux

Introduction Générale

Introduction Générale

L'augmentation du nombre de véhicules circulant dans les grandes villes et l'espace limité pour le stationnement sont des facteurs qui motivent l'adoption de systèmes capables de faire face à de tels problèmes. Les systèmes de stationnement intelligent (Smart Parking) constituent la meilleure solution pour éviter les embouteillages, et la longue recherche d'une place de parking.

Ce travail se concentre sur le développement d'un système de stationnement intelligent à base d'agents. Ce système doit, en particulier, aider les conducteurs à trouver une place de stationnement dans un ensemble de smart parking réalisé à l'aide de la technologie IoT (Internet of Things). D'un outre coté, ce système doit améliorer la gestion des parkings pour les gestionnaires.

Motivation et objectif

Quand on possède un véhicule, trouver une place de parking, dans une ville, est trop souvent source de stress et de temps perdu. De la même façon, lorsqu'on gère un espace de stationnement, on cherche toujours à augmenter le taux d'occupations qui est la clé de la rentabilité, mais souvent trop de places restent vides. Les systèmes de parking intelligent "Smart Parking" ont pour but de faire disparaître ces désagréments, conducteurs comme gestionnaires.

Les Smart-Parkings sont des systèmes de stationnement intelligent, basé généralement sur la technologie d'IoT. Un smart parking peut être considéré comme un objet connecté composé d'une carte à microcontrôleur, des capteurs et des actionneurs pour gérer l'entrée et la sortie des voitures. En utilisant cette technique on peut gérer facilement, et sans intervention humaine, la réservation de places dans un parking. L'utilisation de smart parking dans une ville va permettre par la même occasion de gérer le taux d'occupation des parkings, en signalant celles qui sont libres aux personnes cherchant un stationnement dans une zone.

Malheureusement, en Algérie, ces concepts sont encore très éloignés et l'IoT est encore une question d'imagination. Pour cela, notre objectif dans ce travail est d'aboutir à un modèle multi-agents qui définit un ensemble de règles de fonctionnement, adaptées à la gestion des parkings des villes. Celui-ci est conçu pour aider les conducteurs à trouver une place de stationnement, qui répond à une série de critères, et offre un meilleur service de stationnement pour le public.

L'inclusion des technologies émergentes de l'IOT et des SMA va constituer une approche prometteuse par la réalisation d'un système de stationnement intelligent. Le choix des SMA est motivé par la nature de notre système (ensemble d'individus autonomes (parkings et conducteurs), géographiquement dispersés et voulant se coordonner pour atteindre un objectif) et par les caractéristiques des systèmes multi-agents qui sont devenus un paradigme dominant dans le domaine de développement des systèmes intelligents, distribués et complexes.

Contribution

Dans ce mémoire nous avons développé un système de stationnement intelligent basé sur les technologies de l'IoT et des SMA. Le système réalisé est composé de deux sous-systèmes suivants :

- Le premier, qui joue le rôle d'un gestionnaire de parking, est un système basé IoT réalisé en utilisant une carte arduino avec d'autres composants électroniques. Ce système peut être installer dans les parkings de la ville et permet de gérer l'entrée et la sortie des voitures au parking.
- Le deuxième est un système basé agents qui a pour le rôle de la recherche et la réservation de places de parking, en interagissant avec le premier sous système. Les conducteurs interagissent avec ce système à travers une application mobile pour la rechercher et la réservation d'une place de parking.

Pour la réalisation de notre système, différentes techniques et outils des deux technologies, IoT et SMA ont été utilisés tels que : la plateforme Jade, la méthode voyelle, L'IDE Arduino, etc.

Organisation du mémoire

Le plan du mémoire se subdivisera en cinq chapitres suivants :

Chapitre 1 : dans ce chapitre, consacré à l'IOT, nous commençons par les différentes définitions du IOT, de l'objet connecté et de leurs caractéristiques, ainsi que les différents avantages et les domaines d'utilisation de l'IoT.

Chapitre 2 : ce chapitre présente les notions d'agents et de systèmes multi-agents en déterminant leurs proprietés et leurs caractéristiques. Nous présentons aussi les différentes plateformes de développement des SMA ainsi que leurs principaux domaines d'application.

Chapitre 3 : il concerne la modélisation de notre système smart parking basé sur multi-agents. Dans ce chapitre on va expliquer en détails toutes les phases de l'analyse et la conception de notre système en utilisant la méthodologie voyelle et le langage de modélisation AUML.

Chapitre 4 : c'est un chapitre qui explique la réalisation d'un système de parking intelligent en utilisant ARDUINO.

Chapitre 5 : il concerne la description détaillée de la réalisation et du fonctionnement de notre système de résevation de place de parking. Il comporte aussi une description des différents outils et langages de développement utilisés.

Enfin, nous terminons par une conclusion générale et des perspectives.

Chapitre 1 : Internet des Objets (Internet of Things)

Chapitre1

Internet des Objets

1. Introduction

L'Internet des objets, ou Internet of Things (IoT), a fait le sujet de plusieurs innovations ces dernières années. Aujourd'hui, l'IoT est considéré comme la technologie du futur monde et constitue actuellement un domaine d'investissement rentable dans différents secteurs.L'objectif de ce chapitre est de faire un aperçu général sur l'IoT et les différentes technologies derrières lui.Nous allons aborder le concept d'IoT, son principe de fonctionnement, ses composantes tout en citant des exemples entre autres : quelques protocoles de communication qui lui sont compatibles, les plateformes destinées à leur développement, ainsi que quelques domaines d'application.

2. Définition

Il existe plusieurs définitions de l'IoT dans la littérature, nous citons quelque unes :

Selon le CERP-IdO (cluster des projets européens de recherche sur l'IOT) « l'internet des objets est une infrastructure dynamique d'un réseau global. Ce réseau global a des capacités d'auto-configuration basée sur des standards et des protocoles de communication interopérables. Dans ce réseau, les objets physiques et virtuels ont des identités, des attributs physiques, des personnalités virtuelles et des interfaces intelligentes, et ils sont intégrés au réseau d'une façon transparente » [1] [CLU2010].

L'Union Internationale des Télécommunications définit l'internet des objets comme :« 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 » [2].

La définition de l'IoTrecoupe des dimensions d'ordres conceptuel et technique [2] :

- D'un point de vue conceptuel, l'Internet des objets caractérise des objets physiques connectés ayant leur propre identité numérique et capables de communiquer les uns avec les autres. Ce réseau crée en quelque sorte une passerelle entre le monde physique et le monde virtuel.
- D'un point de vue technique, l'IoT consiste en l'identification numérique directe et normalisée (**adresse IP, protocoles smtp, http...)** d'un objet physique grâce à un système de communication sans fil qui peut être une puce RFID, Bluetooth ou Wi-Fi.

3. Objet connecté

3.1. Définition

De façon simple, un objet connecté est un objet connecté à internet, et qui est capable d'envoyer des informations en temps réel et d'interagir avec son environnement.

De façon plus précise : « un objet connecté est un objet physique équipé de capteurs ou d'une puce qui lui permettent de transcender son usage initial pour proposer de nouveaux services. Il s'agit d'un matériel électronique capable de communiquer avec un ordinateur, un smartphone ou une tablette via un réseau sans fil (Wi-Fi, Bluetooth, réseaux de téléphonie mobile, réseau radio à longue portée de type Sigfox ou LoRa, etc.), qui le relie à Internet ou à un réseau local » [3].

Un objet connecté peut interagir avec le monde physique de manière indépendante sans intervention humaine. Il possède plusieurs contraintes telles que la mémoire, la bande passante ou la consommation d'énergie, etc. Il doit être adopté à un usage, il a une certaine forme d'intelligence, une capacité de recevoir, de transmettre des données avec des logiciels grâce aux capteurs embarqués [ROX 2017].

3.2. Caractéristiques d'un objet connecté

Les caractéristiques d'un objet connecté sont les suivantes [5]:

- ✓ **Identification :** est un code qui lui permet d'être identifié parmi d'autres objets connectés.
- ✓ **Sensibilité à son environnement :** un objet connecté peut avoir la capacité de communiquer avec son environnement.
- ✓ **Interactivité :** la connexion en permanence d'un objet connecté à son réseau n'est pas nécessaire, sauf si l'objet a besoin de communiquer des informations à travers le réseau.
- ✓ **Représentation virtuelle :** est un programme résidant dans le Cloud pouvant agir au nom d'un objet connecté. Cette représentation est nommée parfois cyber-objet ou agent virtuel.
- ✓ **Autonomie :** un objet connecté doit peut fonctionner indépendamment d'un contrôle à distance.

3.3. Composants d'un objet connecté

Les composants de base d'un objet connecté sont : le microcontroleur, les capateurs et les actionnneurs.

3.3.1. Le microcontrôleur

Un microcontrôleur se présente sous la forme d'un circuit intégré qui contient en interne, l'équivalent de la structure complète d'un micro-ordinateur. Un microcontrôleur est composé principalement des éléments de base suivants [6] :

- **Microprocesseur :** C'est l'équivalent du microprocesseur avec une puissance généralement moindre, la vocation n'étant pas la même. C'est cette unité centrale qui exécute le programme et pilote ainsi tous les autres éléments.
- **Mémoire permanente (ROM) :** est une mémoire dont le contenu est conservé même en cas de coupure de courant. Elle contient le programme à exécuter.

- **Mémoire temporaire (RAM) :** elle permettre le stockage des données temporairement qui sont nécessaires à l'exécution du programme de gestion du système technique.
- **Des ports d'entrée/sortie :** permettent au microcontrôleur de communiquer avec son environnement. C'est donc là que vont être connectés les claviers, afficheurs, poussoir, moteurs, relais, etc. que va utiliser l'application.

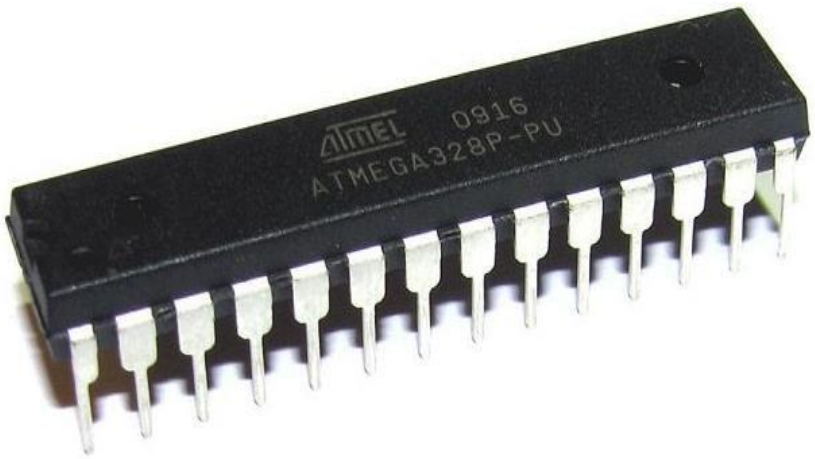


Figure 1.1: Microcontrôleur ATMEL ATmega328P-PU

Il en existe différents modèles de cartes à microcontrôleur pour le développement des projets IoT, nous citons les deux suivants :

- **Cartes Arduino :** ces cartes possèdent un microcontrôleur facilement programmable ainsi que de nombreuses entrées-sorties pour brancher différents devices. Plusieurs cartes Arduino existent et qui se différencient par la puissance du microcontrôleur, par la taille et par le nombre des pins d'entrée/sortie.
  La carte Arduino UNO (Figure 1.2) est la carte la plus couramment utilisée qui constitue un bon choix pour les débutants. L'ensemble des cartes Arduino se programment en C++ à l'aide d'un logiciel de programmation gratuit et open-source fourni par Arduino [7].

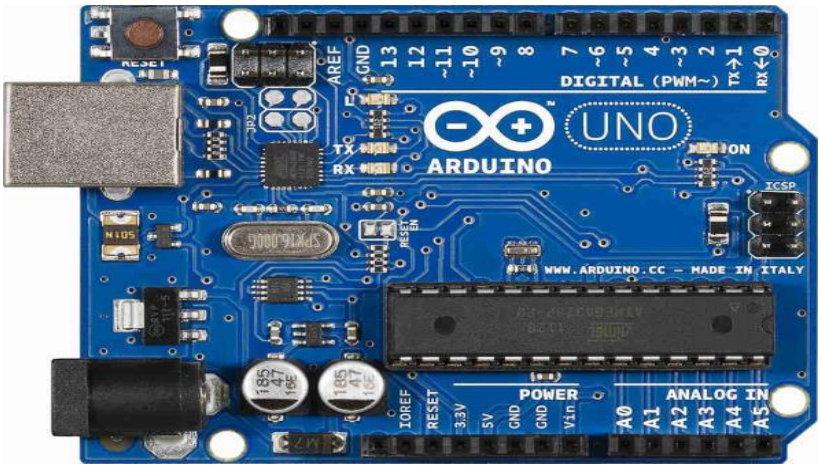


Figure 1.2:Carte Arduino UNO

- **Cartes RaspberryPi :** Le Raspberry Pi est un mini-ordinateur muni d'un processeur ARM, il a été conçu par le créateur de jeux vidéo David Braden. Les cartes RaspberryPi, commeRaspberry Pi 3 (Figure 1.3), sont capables de faire fonctionner un système d'exploitation Linux complet. Il s'agit donc d'un ordinateur bon marché mais avec des fonctionnalités suffisamment puissantes. La carte Raspberry Pi, est essentiellement destinée à des applications d'informatique embarquée.

Figure 1.3: Raspberry Pi 3

3.3.2. Les capteurs

Un capteur est un composant électronique qui permet de transformer une grandeur physique observée en une grandeur électronique [8]. Il existe trois types de capteurs, classifié selon leurs sorties [9]

- **Numérique :** le signal est codé au sein même du capteur par une électronique associée ; ces capteurs sont également désignés par codeurs et compteurs.
- **Logique : (Tout ou rien) :** dont la sortie n'a que deux états électriques, soit Haut (1) soit Bas (0), qui est un cas spécial du type numérique.
- **Analogique :** dont la sortie est un signal continu et proportionnel à la valeur mesurée.

Dans la suite de cette sous section deux exemples des capteurs sont présentés, le premier est un capteur analogique et le deuxième est un capteur numérique.

✓ **Capteur de température LM35** [10]

- Type : analogique
- Voltage : de 4V jusqu'à 30V.
- Intervalle : de -55°C jusqu'à +150°C.
- Fonctionnement : le capteur fournit la température de son entourage en Celsius avec un large intervalle.

Figure 1.4:Capteur de température

✓ **Capteur de mouvements (HC-SR501 PIR)** [11]

- Type : Numérique.
- Voltage : de 4.0V jusqu'à 12.0V (5V recommandé).
- Intervalle : TTL.
- Fonctionnement : le capteur peut détecter le mouvement d'un être-humain ou d'un objet, avec une couverture de 120° comme angle et 7 mètres comme distance.

Figure 1.5: Capteur de mouvements

3.3.3. Les actionneurs

L'actionneur est un dispositif matériel pour transformer une information digitale en un phénomène physique ; d'où sa dénomination. Il peut moduler le comportement ou changer l'état d'un système [12]. Nous citons les deux exemples suivants :

- ✓ **LED3 :** c'est une composante électronique émettant un signal lumineux, généralement blanc, rouge, vert ou bleu.

Figure 1.6: LED 3

- ✓ **Servomoteur** : un servomoteur est un simple moteur électrique, commandé à l'aide d'un servomécanisme. Si le moteur en tant que dispositif contrôlé, associé à un servomécanisme est un moteur à courant continu, il est alors communément appelé servomoteur à courant continu. Si le courant alternatif fait fonctionner le moteur contrôlé, il est appelé servomoteur à courant alternatif [13].

Figure 1.7 : ServoMG90SMikro-Servomotor-Metallgetriebe1x

4. Architectures pour l'IoT

L'architecture IoT [26] peut en fait varier considérablement en fonction de la mise en œuvre. Elle doit être suffisamment ouverte avec des protocoles ouverts pour pouvoir prendre en charge plusieurs applications réseau. Dans la majeure partie des cas elle se compose de 4 blocs constitutifs :

- La scalabilité
- La fonctionnalité
- La disponibilité
- La maintenabilité

Même s'il n'existe pas d'architecture IoT unique universellement acceptée, le format le plus basique et le plus largement accepté est une architecture IoT à quatre couches avec une couche de sécurité (voir figure 1.8).

✓ **Couche de perception** : cette couche est responsable de convertir des signaux analogiques en données numériques et vice versa.

✓ **Couche réseau** : les données collectées par tous ces appareils doivent être transmises et traitées. C'est le travail de la couche réseau, elle connecte donc ces appareils à d'autres objets intelligents, serveurs et appareils réseau. Elle gère également la transmission de toutes les données. Les communications entre les appareils et les services cloud ou les passerelles impliquent différentes technologies : Ethernet, Réseaux cellulaires, LPWAN (Low-power Wide-area Network), WiFi.

- ✓ **Couche de traitement de données** :la couche de traitement accumule, stocke et traite les données provenant de la couche précédente. Toutes ces tâches sont généralement traitées via des plateformes IoT.
- ✓ **Couche Applicative :** La couche application est ce avec quoi l'utilisateur interagit. C'est ce qui est chargé de fournir des services spécifiques à l'application à l'utilisateur.
- ✓ **Couche de sécurité :** Cette couche est transverse à toutes les couches précédentes. La sécurité de l'IoT est primordiale

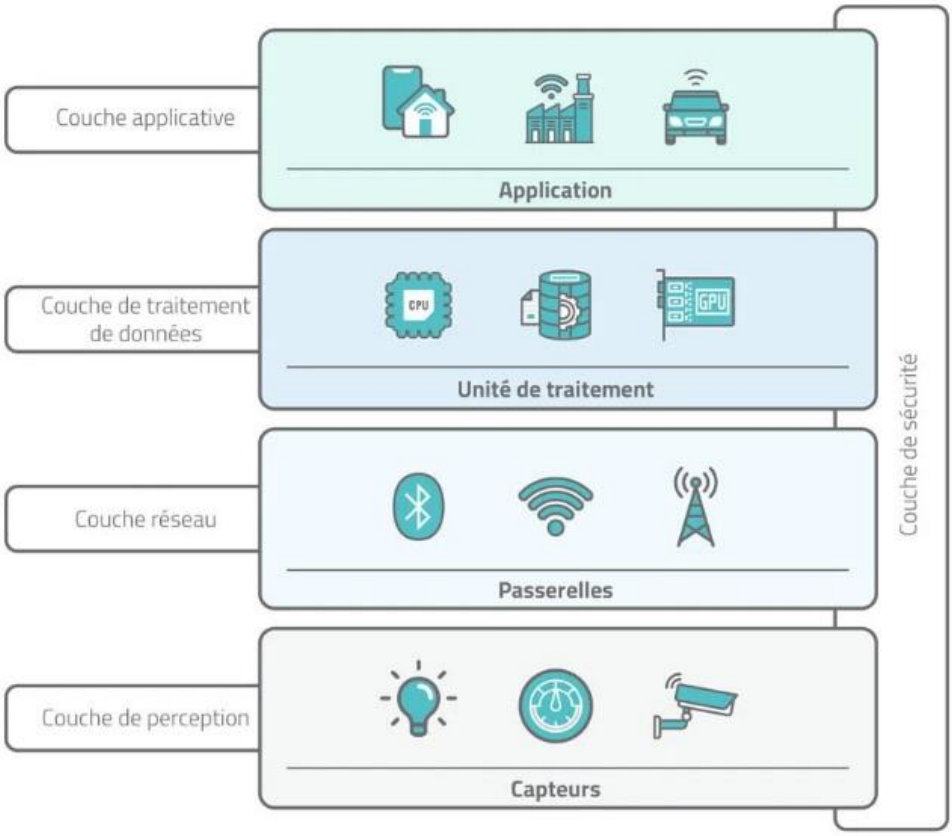


Figure 1.8: Architecture IoT

5. Protocoles de communication

Il existe plusieurs protocoles de communication, assurant l'échange de données entre les différentes parties d'un système d'IoT, nous citons à titre d'exemples [14] :

- **HTTP** (Hypertext Transfer Protocol) : protocole de transfert hypertexte. Ce protocole définie la communication entre un client (exemple : navigateur) et un serveur sur le World Wide Web (WWW).
- **MQTT** (Message Queuing Telemetry Transport) : Il utilise le principe de « Publisher / Subscriber » pour connecter les systèmes entre eux.
- **AMQP** : (Advanced Message Queuing Protocol) : Un protocole pour les systèmes de messagerie orientés messages (MOM).
- **STOMP** : (Simple TextOriented Messaging Protocol) : Est un protocole textuel au-dessus de TCP conçu pour permettre l'interaction avec un middleware orienté messages.

6. Plateformes pour l'IoT

Il existe plusieurs environnements et plateformes de développement pour la programmation des systèmes d'IoT, pouvant être utilisé pour le développement des projets IoT ou pour la collecte et l'analyse de données, nous citons les suivantes :

6.1. L'environement Arduino

L'IDE(IntegratedDevelopementEnvironemente) et un environnement de développement intégré, contenant un éditeur de texte pour écrire du code, une zone de message, une console de texte, une barre d'outils avec des boutons pour les fonctions communes et une série de menus (Figure 1.9). Il se connecte au matériel Arduino pour télécharger des programmes et communiquer avec eux.

Le processus de développement des cartes Arduino suit les étapes suivantes : montage électronique, programmation sur ordinateur, téléversement (téléchargement) du programme vers la carte. Suite au téléversement, le programme s'exécute et le montage électronique produit son œuvre. Le cas échéant, l'utilisateur corrige le montage électronique et/ou le programme [15].

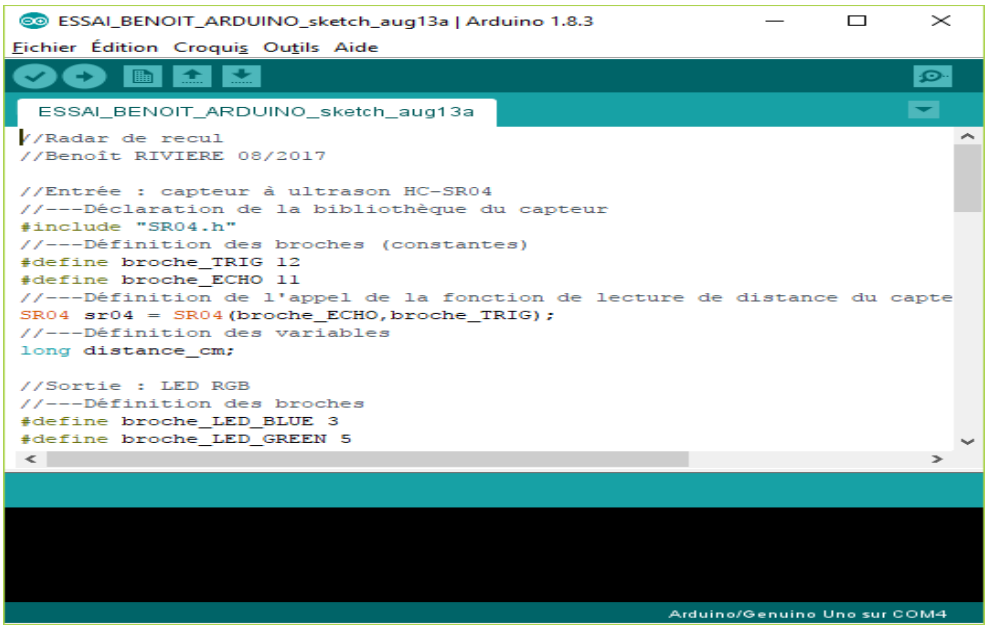


Figure 1.9: L'IDE Arduino

6.2. L'environnement Spyder

Spyder est un environnement scientifique libre et open source écrit en Python, pour Python, et conçu par et pour des scientifiques, des ingénieurs et des analystes de données. Il présente une combinaison unique des fonctionnalités avancées d'édition, d'analyse, de débogage et de profilage d'un outil de développement complet avec l'exploration de données, l'exécution interactive, l'inspection approfondie et les belles capacités de visualisation d'un package scientifique [16].

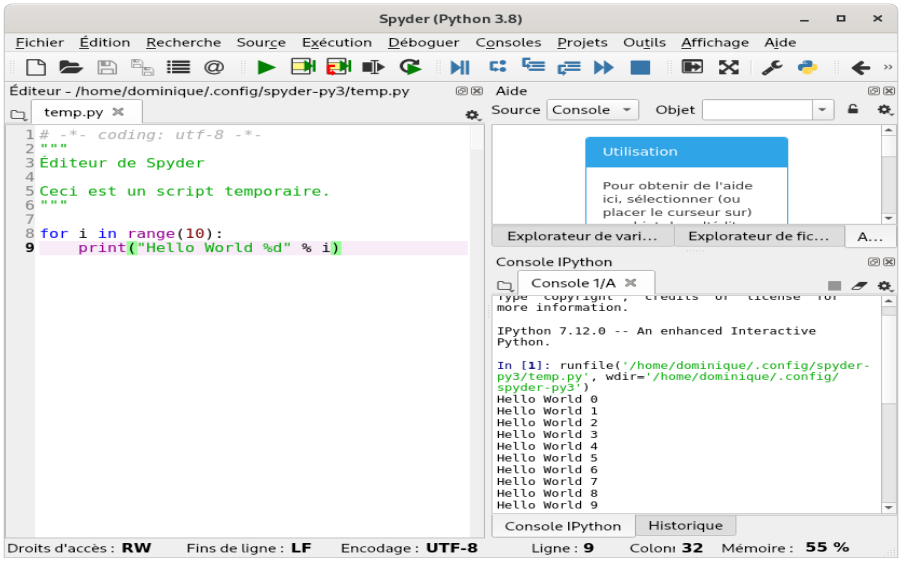


Figure 1.10: L'environnement de développement Spyder

6.3. Cloud9

AWS Cloud9 est un environnement de développement intégré offre une expérience d'édition de code enrichie : il prend en charge plusieurs langages de programmation et débogueurs d'exécution, et comporte un terminal intégré. Il contient un ensemble d'outils utilisés pour coder, créer, exécuter, tester et déboguer des logiciels, et aider à publier des logiciels dans le cloud [17].

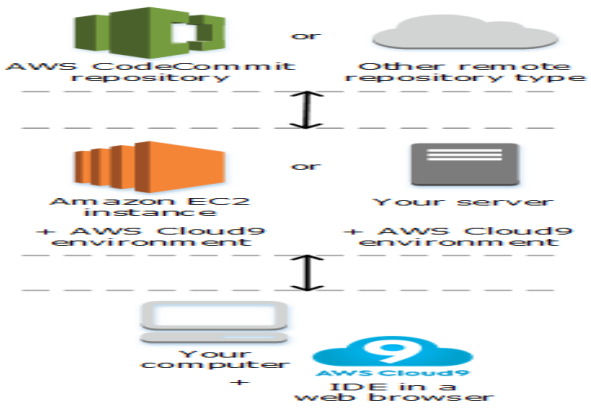


Figure 1.11:L'environnement Cloud9

6.4. Amazon Web Services IoT Platform

AWS IoT (Amazon Internet of Things) est une plateforme Amazon Web Services qui collecte et analyse les données des appareils et capteurs connectés à Internet et connecte ces données aux applications cloud AWS. AWS IoT peut collecter des données à partir de milliards d'appareils et les connecter à des points de terminaison pour d'autres outils et services AWS, ce qui permet à un développeur de lier ces données à une application [18].

6.5. Google Cloud Platform

Est une plateforme de cloud computing fournie par Google, proposant un hébergement sur la même infrastructure que celle que Google utilise en interne pour des produits tels que son moteur de recherche. Cloud Platform fournit aux développeurs des produits permettant de construire une gamme de programmes allant de simples sites web à des applications complexes [19].

7. Domaines d'application d'IOT

Plusieurs domaines d'application sont touchés par l'IoT, Parmi ces principaux domaines nous citons :

7.1. Transports

L'Internet des objets a fait sentir sa présence dans presque tous les secteurs et les transports ne font pas exception. L'IoT dans les transports peut nous aider à suivre les véhicules et les personnes, à connecter les infrastructures et à améliorer le fonctionnement des transports. Internet, ainsi que les applications mobiles, l'augmentation de la communication et de la technologie telles que la collecte de données, la triangulation des téléphones cellulaires, la communication de machine à machine, l'informatique en nuage, etc., le potentiel de révolutionner les transports est élevé [20].

Figure 1.12: IOT et TRANSPORT

7.2. Agriculture intelligente (Smart Agriculture)

L'agriculture intelligente a pour objet de renforcer la capacité des systèmes agricoles, de contribuer à la sécurité alimentaire en intégrant le besoin d'adaptation et le potentiel d'atténuation dans les stratégies de développement de l'agriculture durable.Cet objectif a été atteint enfin par l'utilisation des nouvelles technologies, telles que l'imagerie satellitaire et l'informatique, les systèmes de positionnement par satellite de comme GPS, aussi par l'utilisation des capteurs qui vont s'occuper de récolter les informations utiles sur l'état du sol, taux d'humidité, taux des sels minéraux, etc. et envoyer ces informations au fermier pour prendre les mesures nécessaires garantissant la bonne production[21].

Figure 1.13 : Agriculture intelligente

7.3. E-Santé (e-Health)

Le secteur de la santé a connu un très grand nombre d'applications permettant à un patient et à son docteur de recevoir des informations, parfois même en temps réels, qu'il aurait été impossible de connaître avant l'apparition d'IoT.

Par exemple, (Porteuse Digital Health) qui est le premier **médicament connecté** sur le marché grâce à un capteur directement intégré dans l'être humain qui permet après ça le suivi des patients à distance. On appelle **médicament connecté**, une pilule électronique qui contient un capteur ingérable. Celui-ci est en réalité une version connectée de l'antidépresseur appelé Ability. Le capteur situé dans le comprimé est associé à un autre capteur placé, lui, à l'intérieur d'un patch adhésif collé sur la poitrine. Ce dispositif permet de récolter des données physiologiques suite à la prise de médicaments. Les résultats sont transmis directement au smartphone du patient ou d'un aidant et peuvent être analysés par la suite, par un professionnel de la santé [22].

Figure 1.14 : E-Santé (e-Health)

7.4. La domotique

La domotique regroupe l'ensemble des technologies permettant l'automatisation des équipements d'un habitat. Elle vise à apporter des fonctions de confort : commandes à distance, gestion d'énergie (optimisation de l'éclairage et du chauffage… etc.), sécurité (comme les alarmes) et de communication (contacts et discussion avec des personnes extérieures)

Les services offerts par la domotique couvrent 3 domaines principaux :

1. Assurer la protection des personnes et des biens en domotique par la prévenir des risques d'accident (incendie, fuite de gaz, etc.).
2. Confort de la vie quotidienne surtout pour les personnes âgées ou handicapées
3. Faciliter les économies d'énergie grâce à la réactivité maîtrisée d'une maison intelligente. [23].

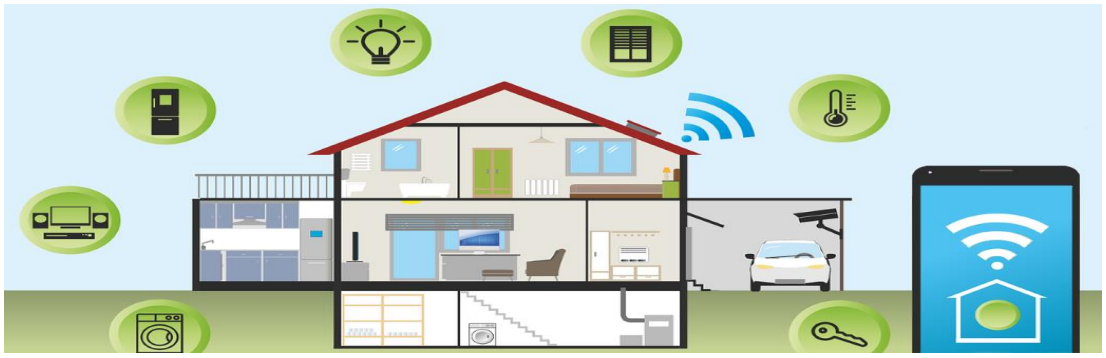


Figure 1.15: La domotique

7.5. Les villes intelligentes (Smart Cities)

Une ville intelligente est celle qui utilise les Technologies de l'Information et de la Communication (TIC) dans différents secteurs d'activité afin d'améliorer la vie quotidienne des usagers et des citoyens. Smart Street Lighting : est un exemple d'application de la ville intelligente. Le système s'active lors de la détection d'un piéton ou un véhicule. Lors de la disparition du mouvement, le système ajuste la luminosité à un niveau minimal optimisé [24].

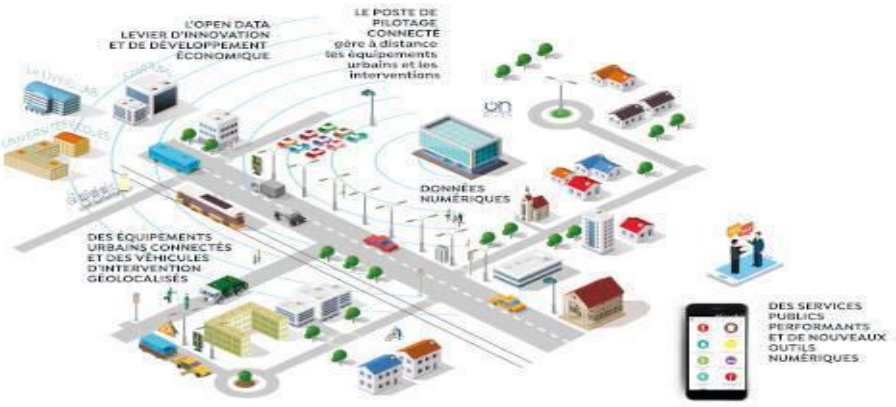


Figure 1.16: Les villes intelligentes (Smart Cities)

7.6. Smart parking

Smart Parking est l'une des solutions Smart City les plus adoptées et à la croissance la plus rapide à travers le monde. Les aéroports, les universités, les centres commerciaux et les garages municipaux ne sont que quelques entités qui ont commencé à réaliser les avantages significatifs de la technologie de stationnement automatisé. La possibilité de se connecter, d'analyser et d'automatiser les données recueillies à partir d'appareils, alimentés par et décrits comme l'Internet des objets, est ce qui rend possible le stationnement intelligent [25].

Figure 1.17: Smart parking

8. Conclusion

Dans ce chapitre, nous avons abordé les concepts clés de l'IoT. Nous avons vu que l'IoT est une nouvelle technologie qui aide à faciliter et améliorer la vie quotidienne des êtres humains en plusieurs domaines. Notre domaine d'application s'articule autour du domaine de smart parking. Nous optons vers le développement d'un système de parking intelligent basé sur l'IoT et le paradigme multi-agents qui fera l'objet du prochain chapitre.

Chapitre 2 : Les Systèmes Multi-Agents

Chapitre2 :

Les systèmes multi-agents

1. Introduction

Le domaine des Systèmes Multi-Agents (SMA) est un axe de recherche très actif. Cette discipline est à la connexion de plusieurs domaines, en particulier de l'intelligence artificielle et de l'informatique distribuée. Cette discipline a vu le jour pour remédier aux insuffisances de l'IA classique et pour permettre la résolution de certains problèmes qui sont distribués de manière inhérente. Les SMA ont pris de plus en plus une place importante parmi les technologies de développement des systèmes complexes et distribués. Cette importance est peut-être reflétée sur l'évolution du marché global consacré aux agents logiciels qui a connu une augmentation importante dans les dernières années.

Dans ce chapitre, nous présentons les principes et les concepts de la technologie agent et des SMA ainsi que leurs caractéristiques, leurs plateformes de développement ainsi quelques exemples de leurs domaines d'application.

2. Agent

2.1. Définition

Dans la littérature, on trouve une multitude de définitions d'agents et il n'existe pas une définition acceptée universellement par les chercheurs concernés. Dans la suite de cette section, nous allons présenter quelques définitions de ce concept.

2.1.1. Définition de Ferber

Selon Ferber [27] : « On appelle agent, une entité physique ou virtuelle

- Qui est capable d'agir dans un environnement.
- Qui peut communiquer directement avec d'autres agents.
- Qui est mue par un ensemble de tendances (sous la forme d'objectifs individuels ou d'une fonction de satisfaction, voire de survie, qu'elle cherche à optimiser).
- Qui possède des ressources propres.
- Qui est capable de percevoir (mais de manière limitée) son environnement.
- Qui ne dispose que d'une représentation partielle de cet environnement (et éventuellement aucune).
- Qui possède des compétences et offre des services.
- Qui peut éventuellement se reproduire.
- Dont le comportement tend à satisfaire ses objectifs, en tenant compte des ressources et des compétences dont elle dispose, et en fonction de sa perception, de ses représentations et des communications qu'elle reçoit ».

La figure (2.1) présente les principales composantes d'un agent et ces interactions avec son environnement.

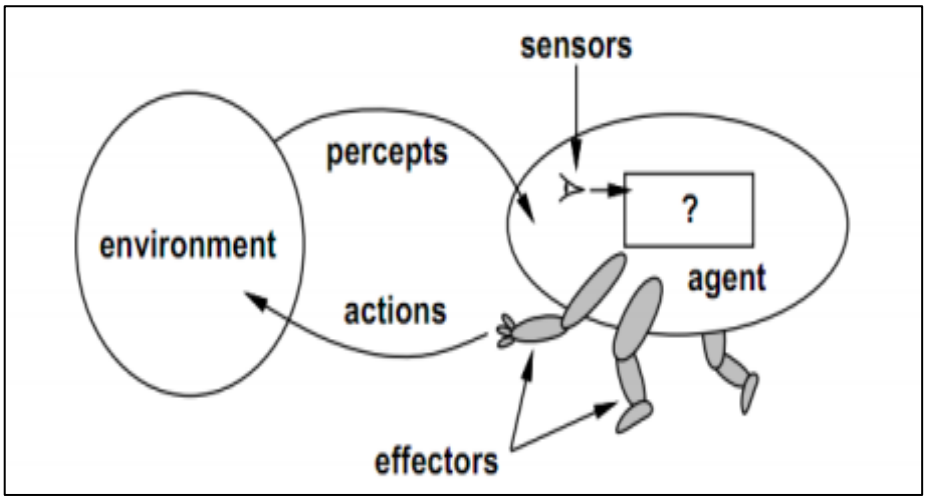


Figure 2.1 : Schéma d'un agent [27]

2.1.2. Définition de RUSSEL et NORVING

RUSSEL et NORVING [28] : « appellent agent toute entité qui peut être considérée comme percevant de son environnement grâce à des capteurs et qui agit sur cet environnement via des effecteurs » (Figure 2.2).

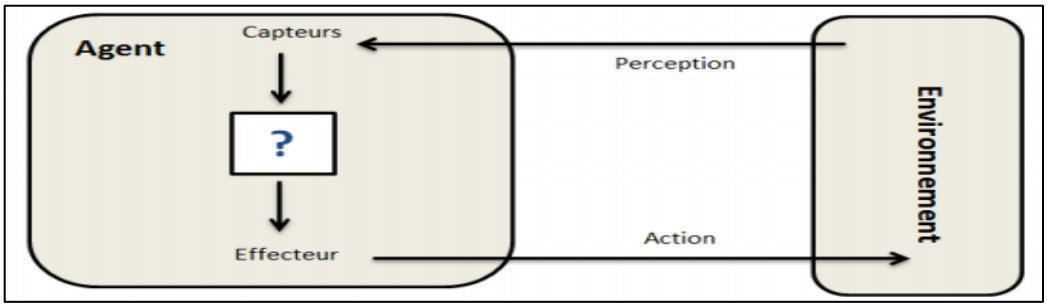


Figure 2.2: Les agents interagissent avec leurs environnements au moyen de capteur et d'effecteurs [28]

2.1.3. Définition de Jennings et al

Jennings, Sycara et Wooldridge [28] définissent un agent comme « un système informatique, situé dans un environnement, et qui agit d'une façon autonome et flexible pour atteindre les objectifs pour lesquels il a été conçu ». Les notions "situé", "autonomie" et "flexible" sont définies comme suit :

- **Situé** : l'agent est capable d'agir sur son environnement à partir des entrées sensorielles qu'il reçoit de ce même environnement. Exemples : systèmes de contrôle de processus, systèmes embarqués, etc.
- **Autonome** : L'agent est capable d'agir sans l'intervention d'un tiers (humain ou agent) et contrôle ses propres actions ainsi que son état interne.
- **Flexible** : L'agent dans ce cas est :
  a) Capable de répondre à temps : l'agent doit être capable de percevoir son environnement et élaborer une réponse dans les temps requis,
  b) Proactif : L'agent doit exhiber un comportement proactif et opportuniste, tout en étant capable de prendre l'initiative au "bon" moment ;

c) Social : L'agent doit être capable d'interagir avec les autres agents (logiciels et humains) quand la situation l'exige afin de compléter ses tâches ou aider ces agents à accomplir les leurs.

2.2. Propriétés d'un Agent

Les principales propriétés d'un agent sont les suivantes [30] :

- **L'autonomie :** Un agent est autonome s'il est capable de prendre des initiatives et d'agir sans l'intervention d'autres agents et notamment de l'être humain.
- **La réactivité :** L'agent intelligent doit réagir dans un temps raisonnable aux changements de l'environnement.
- **La pro-activité :** L'agent intelligent prend l'initiative d'un comportement tendant vers un but précis.
- **Communication** : un agent peut communiquer avec d'autres agents ainsi qu'avec des utilisateurs humains.
- **La sociabilité :** L'agent intelligent doit être capable d'interagir avec d'autres agents intelligents. Il en ressort une caractéristique très importante et propre à un agent intelligent, c'est l'autonomie dans la prise de décision.

2.3. Types d'agents

Nous distinguons principalement trois types d'agents :

2.3.1. Agents réactifs

Les agents réactifs sont des agents avec reflex simple et qu'ayant des comportements du type « stimulus-réponse ». Ce type d'agents ne fait que réagir à des stimulus provenant de son environnement. Ces agents n'ont aucune représentation de leur environnement, aucune base de connaissances et aucune forme de mémoire. Les agents réflexes simples ne possèdent que trois composants (Figure 2.3):

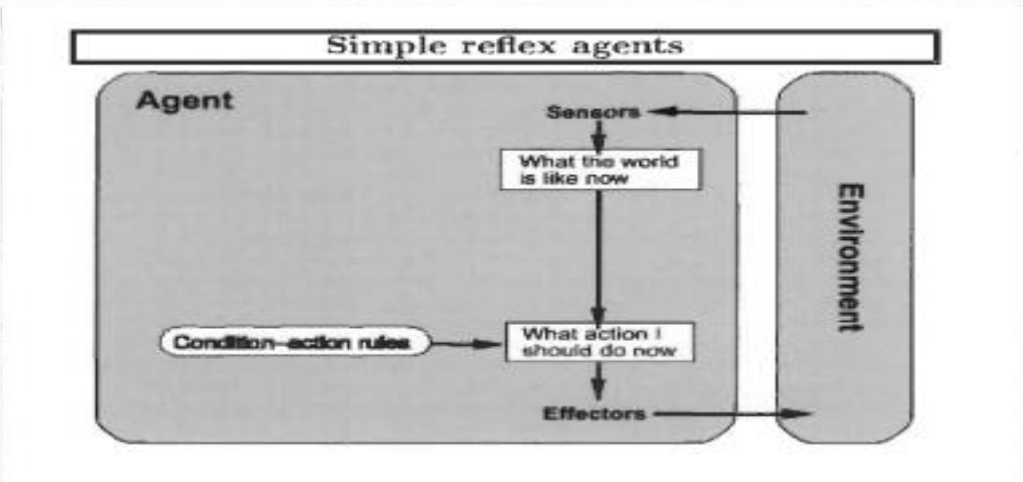


Figure 2.3: Schéma d'un agent réactif [31]

- Les capteurs (senseurs) qui servent à percevoir les changements de leur environnement.
- Un ensemble de règles et de conditions déterminant les actions à effectuer lorsqu'un changement se produit dans l'environnement.
- Les effecteurs qui permettent aux agents d'agir sur leur environnement.

2.3.2. Agents cognitifs

Agents cognitifs sont plus évolués. Ils sont le résultat direct des recherches menées dans le domaine de l'intelligence artificielle sont définis par leur capacité de raisonner sur des représentations du monde. Ils sont capables, à la fois de mémoriser des situations, de les analyser, de prévoir des réactions possibles à leurs actions, d'en tirer des conduites pour les évènements futurs et donc de planifier leur propre comportement [28].

Les agents BDI (Figure 2.4) constitués un exemple très connu des agents cognitifs. Ce type d'agent se fonde sur trois attitudes [27] :

- **Les croyances** : Ce sont les informations que possède l'agent sur son environnement et sur les autres agents agissant sur le même environnement.
- **Les désirs :** Ce sont les objectifs que se fixe un agent et qui motive son comportement.
- **Les intentions** : Ce sont les actions qu'un agent a décidé de faire pour accomplir ses désirs.

Ils forment des ensembles de plans qui sont exécutés tant que l'objectif correspondant n'est pas atteint.

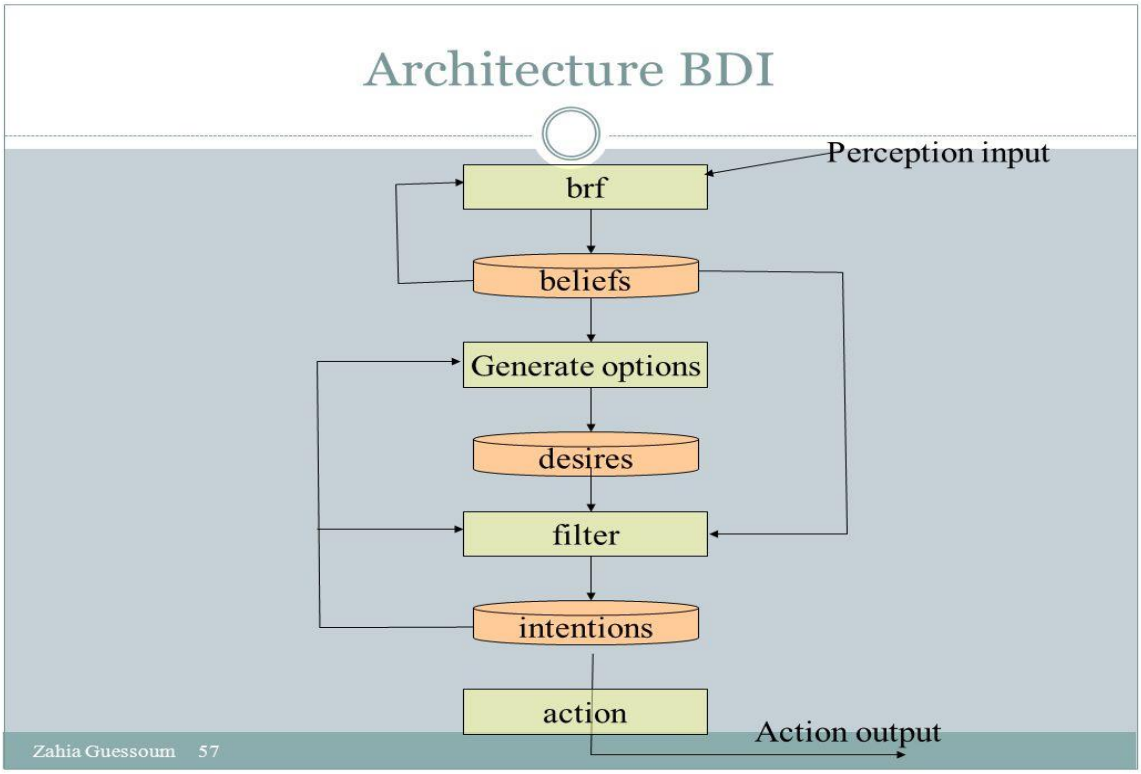


Figure 2.4:Schéma d'un agent cognitif BDI [28]

2.3.3. Agents hybrides

L'agent hybride est une combinaison entre l'agent réactif et l'agent cognitif. Ce type d'agent est composé en plusieurs couches empilées, la plupart des architectures considèrent que trois en suffisent largement. Généralement, la couche au niveau bas est une couche purement réactive, ayant pour rôle de prendre des décisions à partir des données non raffinées perçues de l'environnement. La couche intermédiaire raisonne avec des connaissances de l'environnement et une abstraction des données perçues. La couche supérieure prend en charge les aspects sociaux de l'environnement, c'est-à-dire un raisonnement prenant compte des autres agents [32].

On trouve deux représentations pour cette architecture : horizontale et verticale. La figure (2.5) présente les principales composantes d'un agent hybrides et ces interactions avec son environnement.

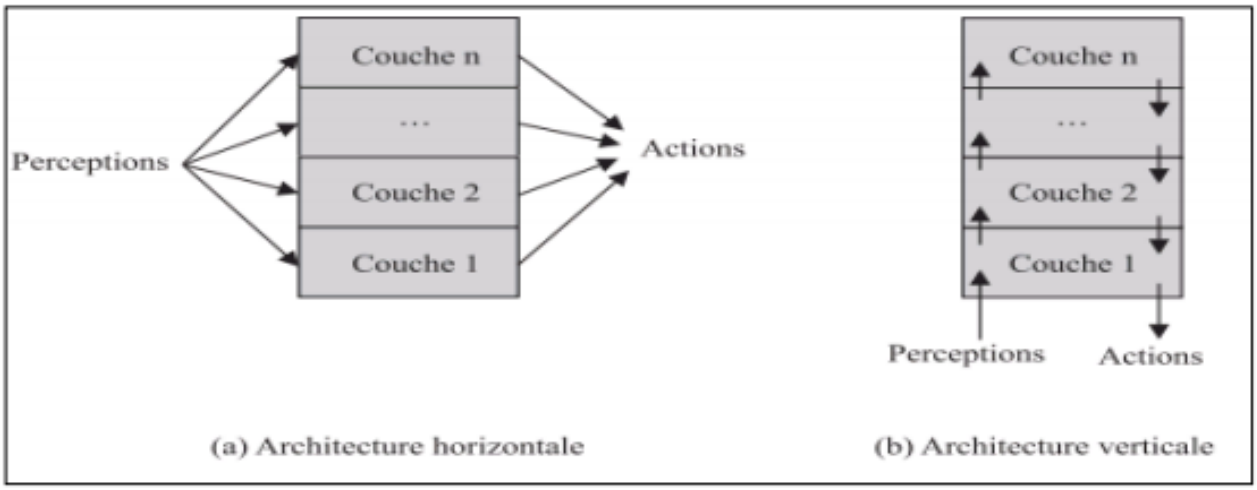


Figure 2.5:Architecture d'un agent hybride [33]

3. Systèmes multi-agents

Les SMA sont des systèmes distribués composés d'un ensemble d'agents situé dans un certain environnement et interagissant selon certaines organisations. Ces systèmes permettent de résoudre des problèmes complexes en exploitant l'intelligence collective des agents qui le compose.

3.1. Définition

Dans la littérature on trouve une multitude de définitions des systèmes multi-agents, parmi elles nous retiendrons celle de Ferber [33], qui le définit comme un système composé des éléments suivants (Figure 2.6) :

- Un ensemble B d'entités plongées dans un environnement E
- Un ensemble A d'agents avec $A \subset B$
- Un système d'action (opérations) permettant à des agents d'agir dans E
- Un système de communication entre Agents
- Une organisation O structurant l'ensemble des agents et définissant les fonctions remplies par les agents

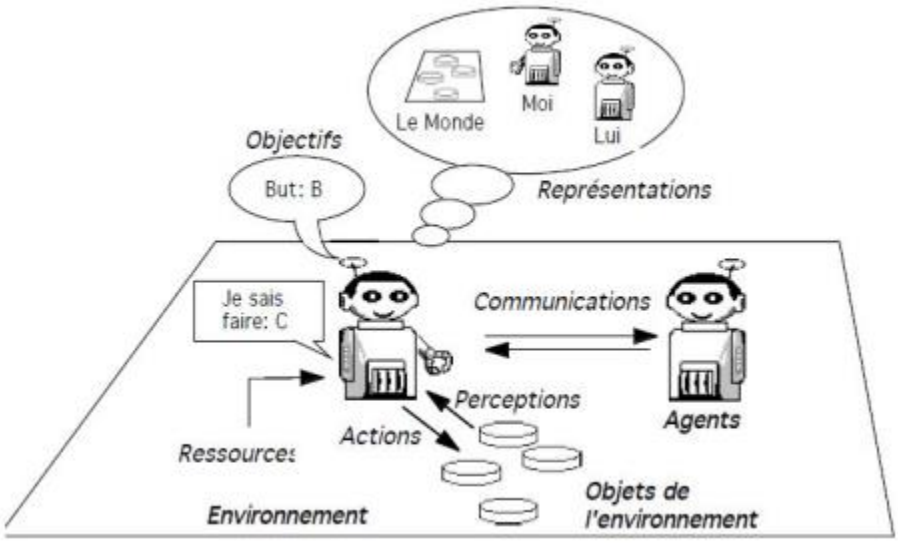


Figure 2.6: Représentation imagée d'un agent en interaction avec son environnement et les autres agents [33]

3.2. Propriétés

Généralement, un SMA possède les propriétés suivantes :

- Chaque agent est une partie du système.
- Le comportement collectif est le résultat de règles sociales et d'interactions et pas d'un surveillant d'autorité centrale (il n'y a pas de décisions centrales valables pour tout le système)
- Chaque agent a un point de vue partiel sur le système pour résoudre un problème
- Un agent est capable de coordonner ses activités avec les autres agents pour accéder à des ressources et à des services partagés dont il a besoin (pour réaliser ses buts)
- Les agents travaillant de façon asynchrone et indépendamment les uns des autres
- Un agent est en activité permanente et prend ses propres décisions en fonction de ses objectifs et de ses connaissances dans le but d'accomplir ses tâches.
- Les agents sont capables de communiquer, d'interagir et une influence sur le comportement entre eux (il n'y a pas d'ordres, seulement des requêtes).
- L'auto-organisation et Adaptation : il est impossible de spécifier le but global et d'organiser les agents pour l'atteindre c'est pourquoi l'auto-organisation et l'adaptation sont considérées des caractéristiques les plus importantes dans les

systèmes multi-agents où le système adapte son comportement à l'environnement en cours de fonctionnement et s'organise sans direction, manipulation ou contrôle externes.

3.3. Architectures SMA

Les SMA Peuvent être conçus selon plusieurs types d'architectures.

3.3.1. Les systèmes centralisés

Dans ces systèmes, on distingue deux classes d'agents : les agents esclaves et l'agent maitre ou responsable (Figure 2.7). Les agents esclaves ne se communiquent pas directement entres eux. Ils envoient et obtiennent les données de l'agent maitre. Les principaux désavantages de ce type d'architecture sont l'inefficacité des systèmes et leur manque de tolérance aux fautes.

Comme toutes les données sont gardées sur la même machine, le manque d'espace mémoire devient souvent un problème. D'un autre côté, lorsque plusieurs agents tentent d'accéder aux mêmes données, un problème de partage des ressources est inévitable [31].

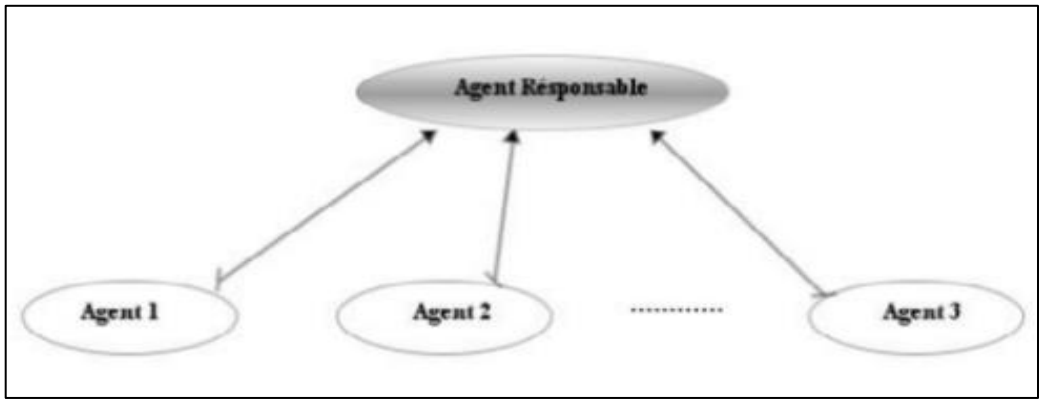


Figure 2.7: Architecture centralisée

3.3.2. Les systèmes hiérarchisés

Plusieurs systèmes utilisent cette architecture pour la réalisation de SMA. Ces systèmes se basent sur une structure où les entités répondent à leurs supérieurs hiérarchiques (Figure 2.8). Ce type de système est relativement simple à implémenter [31].

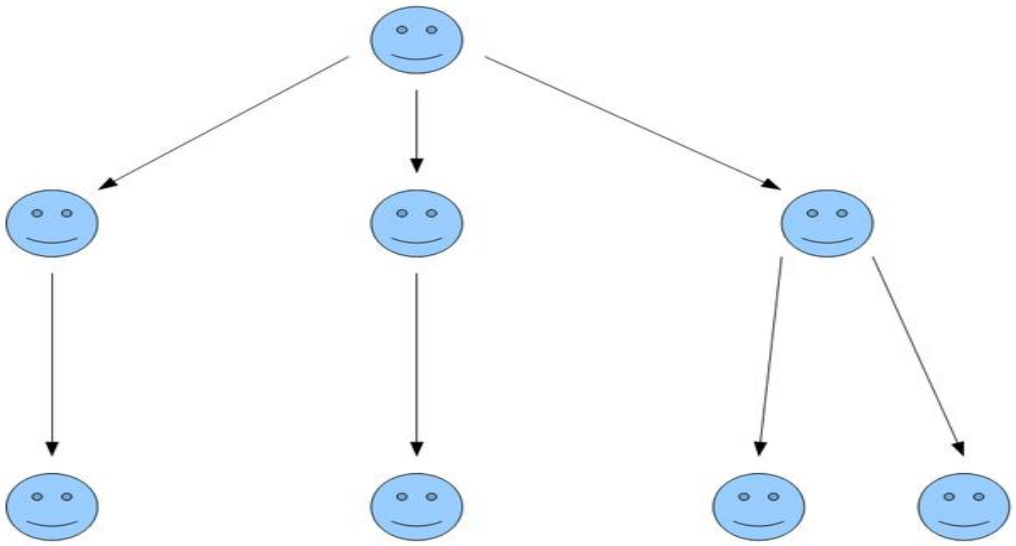


Figure 2.8:Architecture hierarchies

3.3.3. Les systèmes totalement distribués

Cette approche est de loin la plus puissante mais aussi la plus complexe à développer et à implémenter (Figure 2.9). Dans cette architecture tous les agents sont reliés entre eux Si un agent demande un service d'un autre agent, il lui envoie une requête Directe.

L'approche distribuée procure de grands avantages comme la très grande tolérance aux fautes, le partage équitable du travail entre les sous-systèmes et/ou agents, l'utilisation plus uniforme des ressources, l'indépendance et l'autonomie des sous-systèmes, la répartition des tâches, l'efficacité du modèle concurrent et/ou parallèle, leur grande extensibilité, etc. [31].

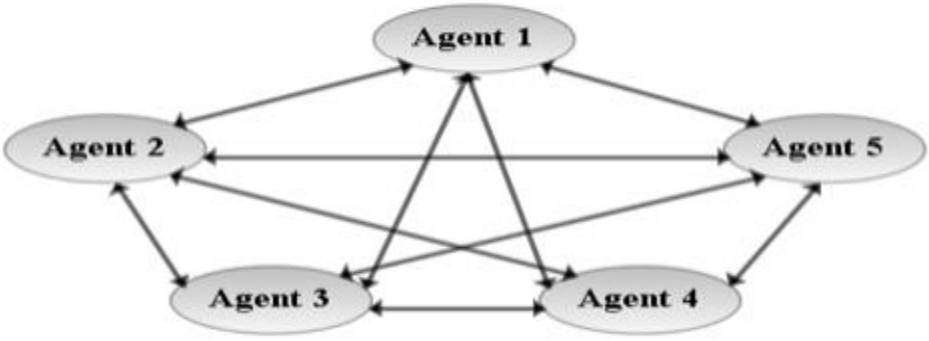


Figure 2.9: Architecture non centralisée

3.4. Interaction entre agents

3.4.1. Définition

La notion d'interaction est au centre de la problématique des SMA. Selon Ferber : « Une interaction est la mise en relation dynamique de deux ou plusieurs agents par le biais d'un ensemble d'actions réciproques. Les interactions sont non seulement la conséquence d'actions effectuées par plusieurs agents en même temps, mais aussi l'élément nécessaire à la constitution d'organisation sociales »[34].

On distingue principalement quatre types d'interactions que les agents peuvent utiliser qui sont : la coordination, la coopération, la communication et la négociation.

3.4.2. Types d'interaction

3.4.2.1. Coordination

La coordination est le processus de construction de programmes en assemblant les parties actives. Ainsi, la coordination détermine en quelque sorte quelles sont les règles de bon fonctionnement du système auquel les agents appartiennent. Selon la nature des agents et des interactions, plusieurs formes de coordination sont possibles. La coordination peut être imposée à un agent par une sorte de contrat, comme le respect d'un protocole d'interaction donné. Elle peut aussi être apprise, lorsque l'agent trouve un intérêt à participer à la collectivité [36].

3.4.2.2. Coopération

La coopération [36] est une caractéristique très importante dans les systèmes multi agents. En effet, une résolution distribuée d'un problème est le résultat de l'interaction coopérative entre les différents agents. Dans le cadre d'une telle dynamique collective, un agent doit disposer en plus de la connaissance reflétant son degré d'implication dans cette dynamique (croyances, buts) d'un certain nombre de compétences nécessaires pour la coopération. Il doit pouvoir :

- Mettre à jour le modèle du monde environnant,
- Intégrer des informations venant d'autres agents,

- Interrompre un plan pour aider d'autres agents.

3.4.2.3. Communication

Le système de communication qui lie un ensemble d'agents agit comme un système nerveux qui met en contact des individus parfois séparés. La communication en effet agrandit les capacités perceptives des agents en leur permettant de bénéficier des informations et du savoir-faire des autres agents. Les communications sont indispensables à la coopération et il est difficile de concevoir un système d'agents coopérants s'il n'existe pas un système permettant aux agents d'échanger des informations ou de véhiculer des requêtes [35].

A. Types

Il existe deux principaux modes de communication : la communication indirecte et la communication directe.

- ✓ **Communication indirecte :**la communication indirecte est la communication qui se fait à travers l'environnement ou une structure de données partagée, autrement dit un agent applique des actions sur l'environnement et un autres perçoivent ces derniers et prend des décisions qui leurs conviennent. Ce type de communication est adapté aux agents réactifs [32].
- ✓ **Communication directe :** la communication directe (Figure 2.10) est une communication qui se base sur l'échange des messages entre les agents. Les messages doivent être formalisés afin d'être compréhensibles par n'importe quel agent [32].

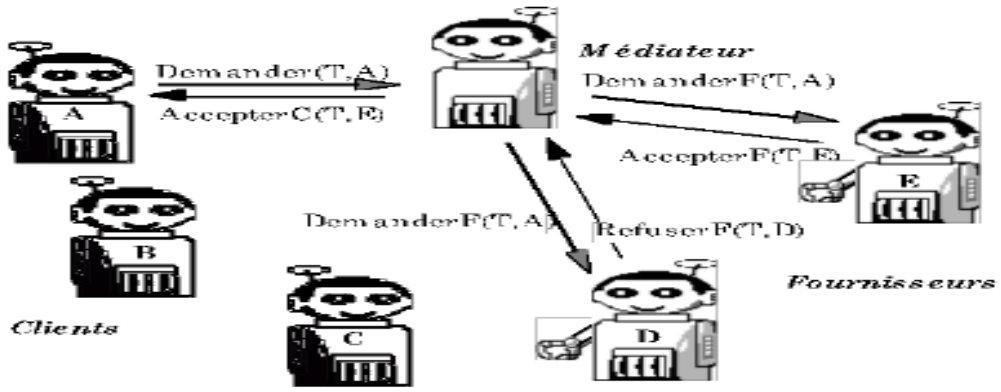


Figure 2.10: Exemple de communication dans système multi-agent (la répartition de tâches par médiation) [32]

B. Langages de communication

Les langages de communication les plus connus dans les SMA sont :

- ✓ **KQML** (KnowledgeQuery and Manipulation Language): **KQML[**32]est un langage de communication inter agents basé sur la théorie de l'acte de langage, ce langage a un ensemble de types de messages prédéfinis appelés 'performatifs' et un ensemble de règles définissant les comportements suggérés pour les agents destinataires de ces messages.
- ✓ **FIPA-ACL** (Foundation for Intelligent Physical Agents – Agent Communication Language) [32] : Le langage ACL de FIPAest fondé sur la théorie des actes de langage qui stipule que les messages représentent des actions, ou actes de communication – également appelés actes de langage ou performatifs, qui est plus riche sémantiquement que le KQML.

3.4.2.4. Négociation

La négociation est une discussion entre deux agents ou plus qu'essaient d'établir une solution à leur problème. Le participant peut être un seul agent, un grouped'agents, un système. La négociation est donc une activité qui consiste à échanger des informations entre participants afin d'aboutir à un compromis mutuellement acceptable [27].

3.5. Plateformes de développement

Les plateformes multi-agents sont des Framework permettant aux développeurs de réaliser des applications sans perdre de temps à réaliser des fonctions de base pour la création et l'interaction entre agents nous allons présenter quelques exemples, les plus connues, des plateformes multi-agents [37] :

- ✓ **JADE (Java Agent Development) :** Un Framework de développement de systèmes multi-agents, open-source et basé sur le langage Java. Il offre en particulier un support avancé de la norme FIPA-ACL, ainsi que des outils de validation syntaxique des messages entre agents basé sur les ontologies.
- ✓ **MAGIQUE :** Une plate-forme pour agents physiquement distribués écrite en Java et fournissant un modèle de communication original d'appel à la cantonade.

Dans MAGIQUE, les compétences sont dissociées des agents. L'architecture des agents et les différentes compétences sont développées séparément. Les compétences sont ensuite greffées comme plugin dans les agents au gré du concepteur. Cette plate-forme est développée au sein du LIFL.

- ✓ **MACE :** Le premier environnement de conception et d'expérimentation de différentes architectures d'agents dans divers domaines d'applications. Dans MACE, un agent est un objet actif qui communique par envoi de messages. Les agents existent dans un environnement qui regroupe tous les autres agents et toutes les autres entités du système. Un agent peut effectuer trois types d'actions : changer son état interne, envoyer des messages aux autres agents et envoyer des requêtes au noyau MACE pour contrôler les évènements internes. Chaque agent est doté d'un moteur qui représente la partie active de l'agent. Ce moteur détermine l'activité de l'agent et la façon dont les messages sont interprétés. MACE a été utilisé pour développer des simulations d'applications distribuées.
- ✓ **SemanticAgent :** Basé sur JADE et permet le développement d'agents dont le comportement est représenté en SWRL. SemanticAgent est développé au sein du LIRIS, il est open-source et sous licence GPL V3.
- ✓ **Jadex :** Une plate-forme agent développée en JAVA par l'université de Hambourg qui se veut modulaire, compatible avec de nombreux standards et capable de développer des agents.
- ✓ **Jagent :** est un Framework open source réalisé en Java dont l'objectif est de faciliter le développement et le test de systèmes multi-agents.

3.6. Domaines d'application

Les systèmes multi-agents étant issus du domaine de l'intelligence artificielle distribuée, sont convenables, généralement, pour le développement des systèmes complexes et les systèmes distribués et hétérogènes où la solution est le résultat de l'interaction de plusieurs entités. En conséquence, plusieurs domaines sont considérés comme des domaines d'application idéals. Bien entendu, il est difficile de citer tous les

domaines d'application des systèmes multi-agents. En fait, nous avons choisi de citer dans ce chapitre seulement quelques exemples de ces domaines [36].

3.6.1. Modélisation des systèmes complexes

Les systèmes complexes sont des systèmes où les techniques de modélisation classique sont difficilement utilisables. En effet, dans ces systèmes, les paramètres sont beaucoup trop nombreux ou contradictoires pour pouvoir être pris en compte. Parfois même, il n'est pas possible de connaître l'ensemble des paramètres qui interviennent pour la modélisation. L'approche multi-agent permet alors d'avoir recours à une modélisation locale. Cette modélisation permet grâce aux principes d'émergence présents dans les SMA d'obtenir un système ayant les propriétés attendues.

3.6.2. Systèmes d'informations coopératifs (SIC)

Les SIC sont généralement caractérisés par la grande variété et le grand nombre de sources d'informations. Ces sources d'informations sont hétérogènes et distribuées soit sur un réseau local (Intranet) soit sur l'Internet. De tels systèmes doivent être capables d'exécuter principalement les tâches suivantes :

- La découverte des sources : Trouver la bonne source de données pour l'interroger
- La recherche d'informations : Identifier les informations non structurées et semi structurées
- Le filtrage des informations : Analyser les données et éliminer celles qui sont inutiles ;
- La fusion des informations : Regrouper les informations d'une manière significative.

Les SMA constitués un moyen très efficace pour le développement des systèmes d'informations coopératifs. À titre d'exemple, sur l'utilisation des SMA dans ce domaine, le système NETSA. Ce système est un système multi-agent coopératif, développé à l'université Laval, et destiné aux environnements riches en informations.

3.6.3. Systèmes d'aide à la décision :

Les systèmes d'aide à la décision (SAD) sont présents dans de nombreux domaines et ont pour objectif d'aider le décideur dans sa tâche en lui fournissant tous les éléments pertinents pour la prise de décision. Cela consiste très souvent à extraire de l'information depuis de multiples sources et à la traiter. Des traitements de l'information pour une prise de décision multicritères sont alors nécessaires. Les systèmes multi-agents apparaissent comme étant bien adaptés pour traiter de l'information qui peut revêtir diverses formes et provenir de diverses sources. En effet, il faut pouvoir faire la corrélation entre l'ensemble des éléments obtenus pour les présenter à l'utilisateur. L'approche à base d'agents permet d'effectuer cette corrélation en utilisant la négociation et la coopération entre agents. Il s'agit donc d'un domaine d'application pour les SMA.

3.6.4. Commerce électronique et les agents du Web :

Le domaine du commerce électronique est un domaine en plein essor qui permet de favoriser les transactions commerciales. Ce domaine utilise l'outil informatique et plus particulièrement les ressources mises à disposition par Internet pour rapprocher les acteurs commerciaux dans certains domaines. Ce domaine se rapproche de celui de l'aide à la décision et peut même être confondu avec celui-ci dans certaines circonstances car il est caractérisé par la mise en place de moyens permettant d'extraire de l'information sur les produits, les marchés et les acteurs du marché.

4. Motivation d'utilisation du paradigme multi-agents

Les systèmes multi agents sont des systèmes pouvant représenter des problèmes complexes à plusieurs méthodes de résolution. Ces systèmes possèdent les avantages de la résolution distribuée et concurrente de problèmes, les avantages des systèmes multi agents qui nous ont motivé choisir ce paradigme sont les suivants [38] :

- **La modularité :** la modularité du système est due essentiellement à l'autonomie des agents. Comme les interactions inter agents se basent essentiellement sur l'échange de messages, le couplage entre eux est faible. Ainsi d'autres concepts de base des systèmes multi agents contribuent à cette propriété.

- **La réutilisation :** la réutilisabilité est due à la modularité du système.
- **La facilité de maintenance :** cet avantage est vérifié grâce à la modularité du système.
- **La fiabilité :** la fiabilité est la capacité d'un système de continuer à fonctionner malgré l'apparition des pannes ou des erreurs, cet avantage est vérifié grâce à l'autonomie des agents.
- **L'efficacité :** L'efficacité des logiciels est mesurée en fonction des ressources utilisées au cours du processus de la résolution du problème. Comme les systèmes multi agents sont des systèmes distribués avec des ressources distribuées, ces dernières sont utilisées de façon rationnelle.
- **L'adaptation de réalité :** grâce aux caractéristiques des agents, ce paradigme permet de modéliser des phénomènes réels.
- **Les modes d'interaction sophistiqués :** ce paradigme supporte des modes d'interaction sophistiqués par rapport au paradigme d'objet, comme la coopération, la coordination et la négociation.

5. Conclusion

Les SMA proposent aujourd'hui une nouvelle technologie effective de mise en œuvre de systèmes complexes dès lors que ceux-ci sont dotés, entre autres, de distribution, d'ouverture, d'intelligence, de coopération et d'autonomie. Dans ce chapitre, nous avons présenté un bref survol sur les systèmes multi agents, en expliquant quelques notions clés de ce vaste domaine, cela pour motiver son utilisation et montrer son importance dans notre futur système. Le chapitre suivant sera consacré à l'analyse et la conception de notre futur système.

Chapitre 3 : Analyse et conception

Chapitre 3 :
Analyse et conception

1. Introduction

Après avoir présenté les concepts de système multi-agents et de la technologie IOT dans les chapitres précédents, nous allons utiliser les concepts des systèmes multi agents pour assurer les différentes activités de cycle de vie de notre système.

Dans la première section de ce chapitre, consacré à l'analyse et la conception de notre système, nous allons faire une description fonctionnelle du système ainsi que la méthodologie de développement suivie. Ensuite, nous allons présenter la méthode voyelle et le langage de modélisation AUML (Agent UnifiedModelingLanguage) que nous avons utilisé pour la modélisation de notre système. Dans la troisième section, nous allons identifier les agents et les différentes interactions entre ces derniers et en nous terminerons par la partie conception ou nous allons présenter les différents diagrammes décrivant notre application.

2. Description du système à réaliser

Le système que nous voulons réaliser est un système de parking intelligent basé sur les technologies IoT et agent pour le but de faire disparaitre le stresse de chercher une place de stationnement et ne pas perdre le temps à rechercher le plus proche parking pour les conducteurs.

De l'autre côté, le système permet aux gérants des espaces de stationnement d'éviter que trop de places reste vides (sans occupation) et donc obtenir un fort taux d'occupation qui est la clé de la rentabilité. Donc les utilisateurs de ce système peuvent être des conducteurs (clients), des gestionnaires des parkings et l'administrateur de système.

La figure (3.1) donne une vue globale sur notre système. Le système va se composer des deux sous-systèmes suivants :

- **Smart Parking sous-système** : qui joue le rôle d'un gestionnaire de parking, est un système basé IoT réalisé en utilisant une carte arduino avec d'autres composants électroniques. Ce système est installé dans chacun de parkings de la ville et permet de gérer l'entrée et la sortie des voitures au parking. Nous discuterons la réalisation de ce sous-système dans le chapitre suivant.
- **Sous-système de recherche et de réservation** : est un système basé agents (SMA) qui a pour rôle la recherche et la réservation des places de parking, en interagissant avec le premier sous système. Les conducteurs interagissent avec ce système à travers une application mobile pour la rechercher et la réservation d'une place de parking. Le reste de ce chapitre présente les étapes d'analyse et de conception de ce système de réservation.

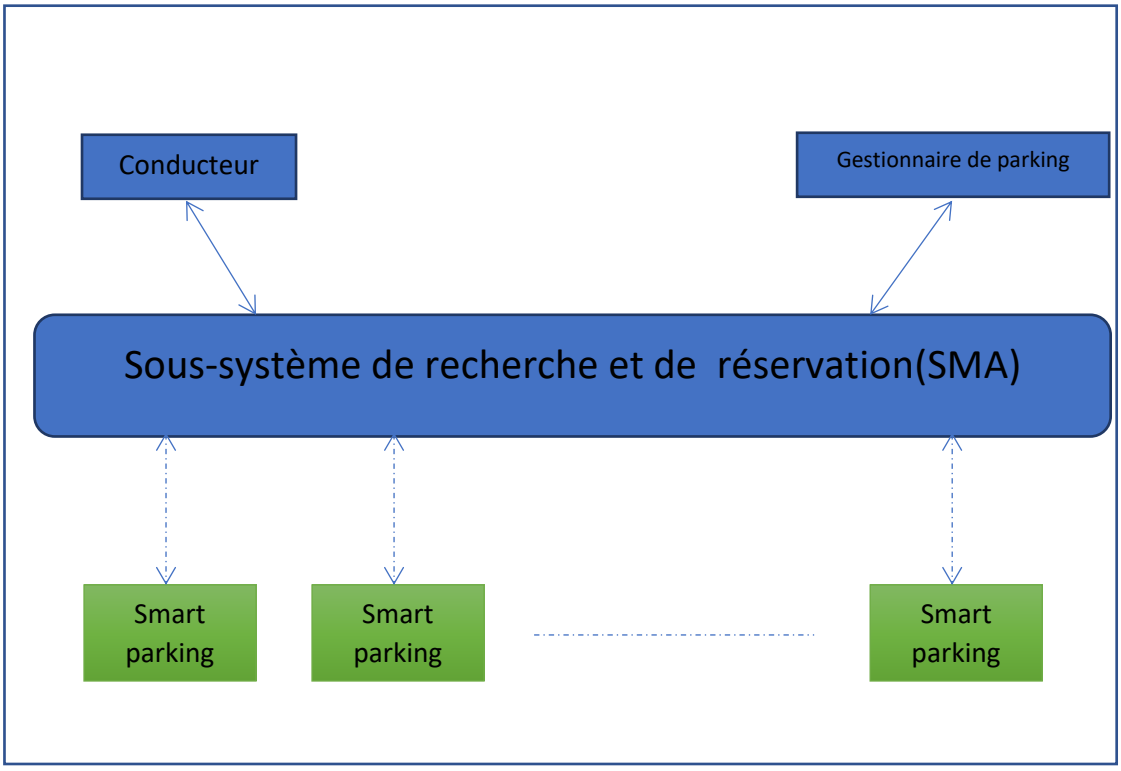


Figure 3.1: Vue globale sur le système de stationnement intelligent

3. Choix méthodologique

3.1. Méthode Voyelle

Les systèmes multi-agents (SMA) sont actuellement très largement utilisés, particulièrement pour les applications complexes nécessitant l'interaction entre plusieurs entités. Dès lors, il est devenu impératif de s'investir dans les méthodologies orientées-agent et autres techniques de modélisation adéquates.

Parmi les méthodes qui couvrent le mieux le cycle de développement d'un système multi-agents, nous avons choisissons la méthodologie Voyelles pour modéliser notre système. Ce choix se justifie par les caractéristiques suivantes de cette approche :

a) C'est une approche qui incite à penser à la conception du système simultanément en termes de cinq dimensions (Figure 3.2) : Agent, Environnement, Interaction, Organisation et Utilisateur [Boissier et Demazeau, 1996; Ricordel et Demazeau, 2000, 2002; da Silva et Demazeau, 2002].

- Les agents : ils concernent les modèles (ou les architectures) utilisés pour la partie active de l'agent, d'un simple automate à des cas plus complexes, comme des systèmes à base de connaissances.
- Les environnements : ce sont les milieux dans lesquels sont plongés les agents. Ils sont spatiaux dans la plupart des applications proposées.
- Les interactions : ils concernent les infrastructures, les langages et les protocoles d'interaction entre agents, depuis de simples interactions physiques à des interactions par actes de langage.
- Les organisations : qui structurent les agents en groupes, hiérarchies, relations...
- Utilisateurs : cette dernière a été créée récemment pour définir les utilisateurs de système.

Figure 3.2: Représentation d'un système multi-agents avec la méthodologie voyelle

b) Voyelles n'est couplée à aucune notation ni plateforme, ce qui offre la possibilité d'utiliser le langage AUML (Agent Unified Modeling Language : cf. Annexe B), qui est une extension du langage UML, dans la phase de conception du système et la plateforme Jade pour réaliser le système.

c) Voyelles repose sur des principes purement multi-agents.

Le processus de développement avec la méthode Voyelles comporte trois étapes essentielles [39] :

- **Analyse :** consiste à identifier les cinq composantes suivantes : Agent(A), Environnement(E), Interactions(I), Organisation (O), Utilisateurs (U).
- **Conception :** il s'agit de choisir les modèles opérationnels des composantes.
- **Implémentation :** consiste en l'instanciation des modèles en utilisant des plateformes et des langages choisis.

3.2. Langage de modélisation AUML

AUML est un langage de modélisation graphique qui a été standardisé par FIPA (Foundation for Intelligent Physical Agents). Il a été proposé comme une extension du langage de modélisation unifié (UML). Jusqu' à maintenant il n y'a pas de standard reconnu pour la modélisation des systèmes multi-agents et AUML a émergé comme un candidat pour assumer une telle position. Il utilise les caractéristiques de "décomposition", 'd'abstraction", et "d'organisation", qui réduisent la complexité de développement de logiciel.

AUML décompose le système en petites parties d'objets, de modèles, de cas d'utilisation ou de classes, et d'actions opérationnelles. Concernant "l'abstraction", elle offre une vue spécialisée abstraite de la modélisation (classe, cas d'utilisation, diagramme, interface, etc.).

Agent UML est une notation de soutien pour les systèmes de développement orientés agent. Il consiste à utiliser le langage de modélisation UML et de son extension afin de représenter les agents, leurs comportements et leurs interactions entre eux. Elle

offre généralement quelques Framework (classe, diagramme, interface, etc.) pour voir comment les agents peuvent être construits dans un système.

La partie principale d'AUML est les mécanismes de modélisation des protocoles d'interaction des systèmes multi-agents. Cela est réalisé par l'introduction d'une nouvelle classe de diagramme à UML : "diagramme de Protocole". Ces diagrammes étendent les diagrammes de séquence en incluant : les rôles d'agents, les comportements des agents, les interactions entre agents, etc.

3.2.1 Les diagrammes AUML

Agent UML est une extension d'UML donc il hérite des représentations proposées par UML. Les différents diagrammes sont :

1. Diagrammes de séquence
2. Diagramme de collaboration
3. Diagramme d'activité
4. Diagramme d'état transition
5. Diagramme de cas d'utilisation
6. Diagramme de classe
7. Diagramme d'objets
8. Package
9. Diagramme de composants

4. Analyse

Cette étape consiste à identifier les cinq composantes suivantes : Agent (A), Environnement (E), interactions (I), Organisation (O), Utilisateur(U). Nous allons commencer d'abord par l'identification des utilisateurs de notre système. Ensuite nous allons passer à identification des autres composantes.

4.1. Identification des utilisateurs

Dans notre système, nous identifions trois types d'utilisateurs : Les conducteurs, les gestionnaires des parkings et l'administrateur.

Les conducteurs : ce sont les personnes qui accèdent à l'application pour chercher et réserver une place de stationnement. Parmi les activités principales des conducteurs dans le système on trouve :

- La recherche et la consultation des places de stationnement proposées par les différents gestionnaires des parkings.
- La réservation en ligne (commande de réservation en ligne, payement, etc)
- La consultation de son compte et de toutes les informations concernant ses réservations.

Les gestionnaires des parkings : ce sont les gérants des parkings qui utilisent notre système pour proposer leurs services. Parmi les activités principales des gestionnaires des parkings dans le système on trouve :

- La création des parkings.
- La mise à jour des parkings.
- Consultation des informations concernant son compte et ses réservations.

L'administrateur : c'est le responsable de la gestion de l'application. Les activités principales de l'administrateur dans le système sont:

- Gérer les comptes des utilisateurs : valider les comptes des gestionnaires des parkings et supprimer et désactiver des comptes des utilisateurs.

4.2. Identification des agents

Le système est composé de plusieurs types d'agents, chacun joue un rôle bien précis. Comme il est montré dans la figure (3.3), les agents de notre système s'exécutent dans le serveur principal sauf les deux agents conducteur et parking qui va s'exécuter sur les appareilles des utilisateurs.

Les agents de notre système sont des agents reactifs sauf l'agent de reservation qui est un agent cognitif, il possede un degré d'intelligence pour rechercher le prking le plus proche au conducteur.

Dans la suite nous allons présenter les différents agents de notre système ainsi que les rôles joués par chacun.

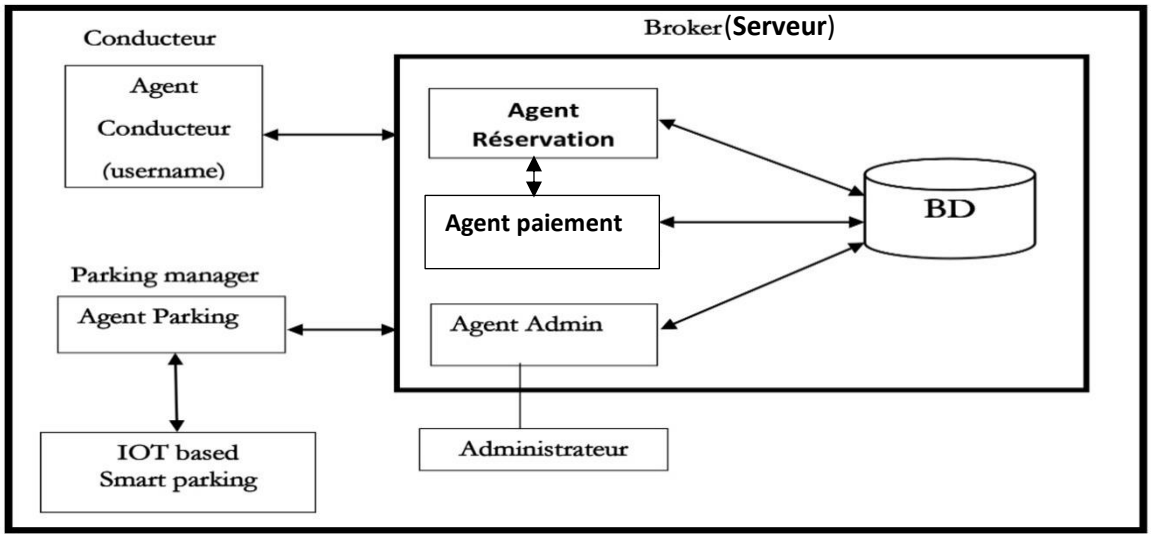


Figure 3.3: Architecture générale du système

4.2.1. Les agents utilisateurs

- **Agent Conducteur :** c'est un agent réactif qui s'exécute dans l'appareil du conducteur, Il permet à l'utilisateur de s'interagir avec notre système. Cet agent est doté d'une interface graphique permettant de fournir au conducteur toutes les l'information nécessaires pour effectuer les opérations de réservation (Le résultat de la recherche, les informations sur les places de parking. . . etc.).
- **Agent Parking :** c'est un agent réactif qui s'exécute dans l'appareil du gestionnaire de parking, Il permet à l'utilisateur de s'interagir avec notre système. Cet agent est doté d'une interface graphique permettant de fournir au gestionnaire toutes les l'informations nécessaires pour effectuer les opérations de gestion et de mise a jour (ajout de parking, les informations sur les places de parking, , . . . etc.). en outre, il est responsable de la gestion intelligente de parking.

4.2.2. Les agents systèmes (Broker)

- **Agent Réservation :** c'est un agent cognitif, il est responsable des opérations (rechercher, demande de réservation, consulter, ...). IL reçoit des requêtes à partir de l'agent conducteur et prend un contact avec les autres agents pour répondre à ces requêtes et fournit des meilleures propositions pour le conducteur.

- **Agent Admin :** il représente l'entité virtuelle d'administrateur au sein du système et il est le responsable des opérations d'administrateur dans le système ainsi que la création de tous les autres agents du système et les opérations de l'inscription et l'authentification.
- **Agent paiement :** est un agent reactif intermédiaire entre l'agent Conducteur et l'agent de Parking, il interagit avec l'agent de reservation pour réaliser l'opération de paiement.

4.3. Environnement

Dans les SMA l'environnement d'un agent est constitué des agents du système et des autres entités physique ou logique avec les quels il s'interagit, comme il est montré dans la figure (3.4).

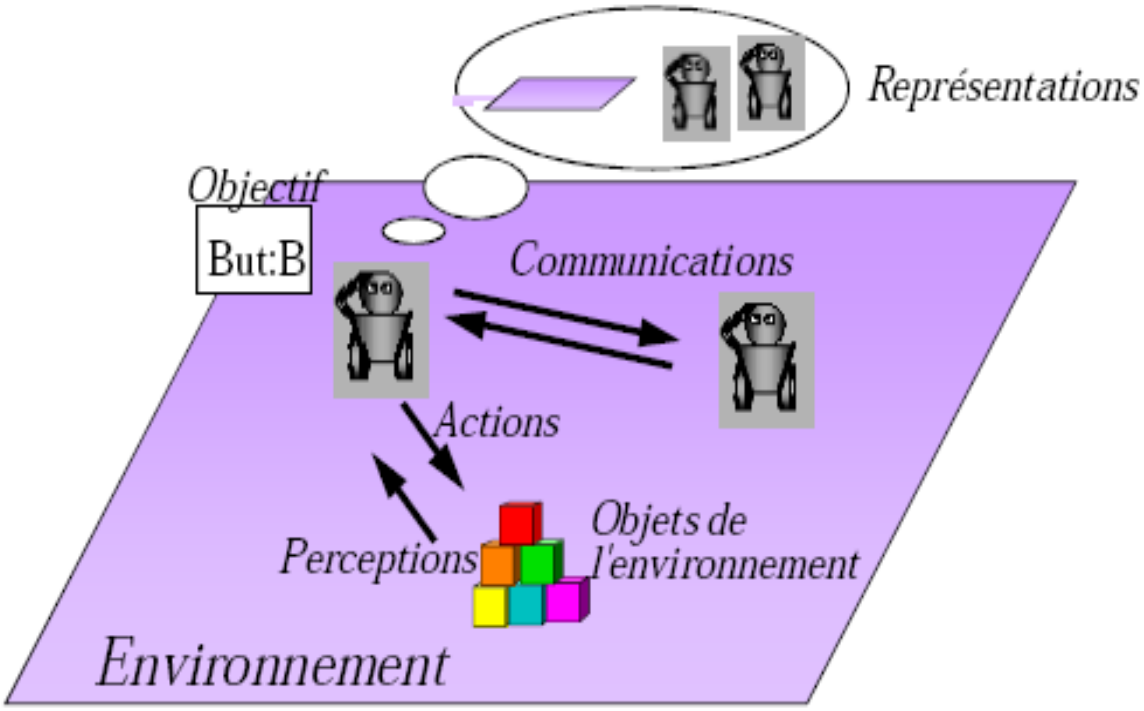


Figure 3.4: Environnement SMA

Le tableau (3.1) sous-dessous montre l'environnement de chacun des agents de notre système ainsi que les agents et les objets avec qui il intéragit.

Agent	Environnement	Agents et Objets avec qui il interagit
Agent conducteur	- Le conducteur et son appareil	- Agent Paiement - Agent Réservation - Agent Admin
Agent parking	-Gestionnaire de parking et son appareil -Système de gestion de parking intelligent basé IOT	- Agent Réservation - Agent Admin
Agent Réservation	- Serveur	- Agent parking - Agent conducteur - Agent paiement - Base de données de système
Agent administrateur	- Serveur	- Agent parking - Agent conducteur - Base de données de système
Agent paiement	- Serveur	- Agent parking - Agent conducteur - Agent réservation - Base de données de système - Banque

Tableau 3.1: Environnement des agents de notre système

4.4. Identification des interactions

Après l'identification de différents agents constituant notre système, nous passons dans cette étape à l'identification des interactions entre ces agents.

- **Interaction Agent Conducteur/Agent Admin :** L'agent conducteur demande l'authentification à l'agent Admin, ce dernier crée un agent Réservation lié à ce dernier.
- **Interaction Agent Conducteur/Agent Réservation :** L'agent Conducteur peut envoyer une demande de recherche d'une place de parking ou une demande de réservation à l'agent de réservation, ce dernier effectuée les opérations et renvoie les résultats à l'agent Conducteur.

- **Interaction Agent Réservation/Agent Parking :** L'Agent Réservation peut envoyer à l'agent Parking une requête de recherche d'une place libre dans son parking ou une requête pour accomplir l'opération de réservation.
- **Interaction Agent Admin/Agent Parking :** Passer les informations de mise à jour de parking.
- **Interaction Agent Paiement/Agent Réservation :** L'agent réservation envoie les informations nécessaires pour effectuer le paiement et l'agent paiement répond avec réussi ou échoué.
- **Interaction Agent paiement/Agent Parking :** Agent paiement envoie les informations de réservation (réussi ou échoué) a l'agent Parking pour faire la mise à jour de la demande.

4.5. Organisation

L'organisation est une structure décrivant comment les agents dans l'environnement sont en relation et interagissent afin d'atteindre des buts. L'organisation structurelle est :

- Côté des trois utilisateurs du système on retrouve GUI pour chaque acteur qui relie chaque GUI avec l'agent correspondant à l'acteur lors de connexion au système : conducteur, gestionnaire de parking et administrateur.
  Les GUI du système représentent ses points d'interaction avec les utilisateurs du système.
  La page d'accueil de l'utilisateur lui permet de se connecter ou de s'inscrire, une fois que les utilisateurs s'est connecte l'agent Admin lui crée un agent Conducteur, agent Parking, agent Réservation).
- Coté serveur on trouve un Agent Admin et un Agent paiement et un agent réservation. L'agent Admin est responsable du système (inscription, authentification, gérer application), l'agent de réservation est responsable du processus de recherche et de réservation, et l'agent paiement est responsable du processus de paiement.

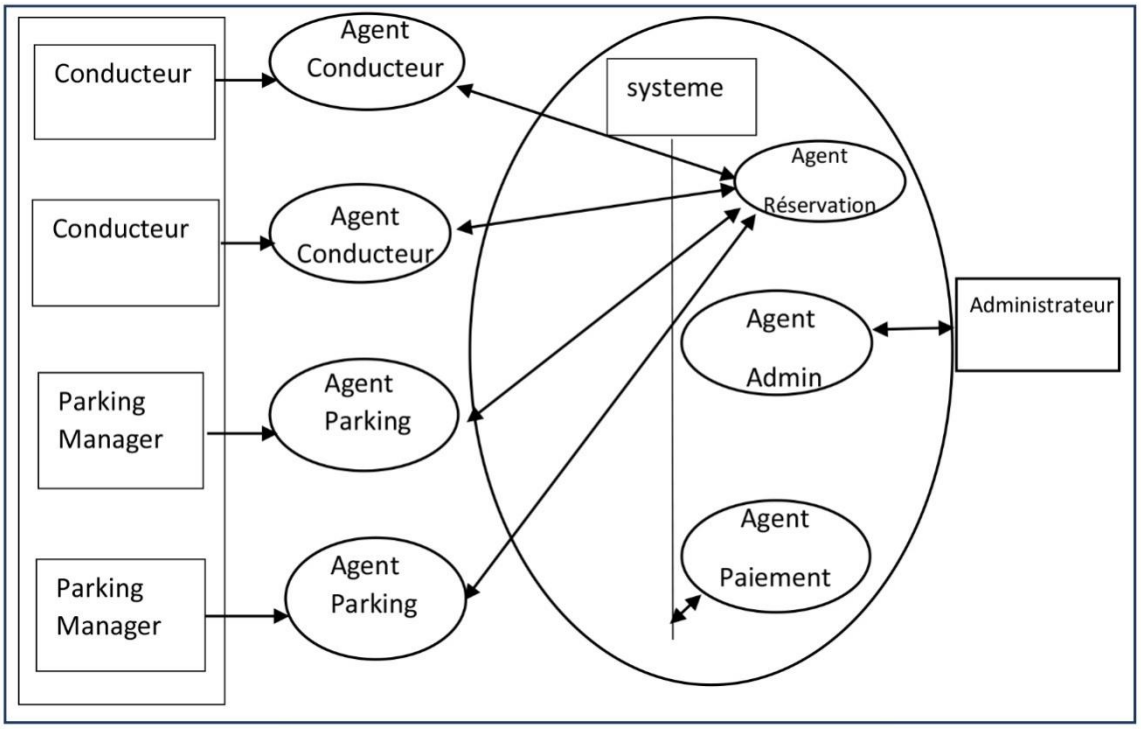


Figure 3.5: Organisation du système

5. Conception :

Pour la modélisation de notre système on utilise le langage AUML parce que dans l'approche voyelle il n'exige pas une notation pour la modélisation de système. Notre conception va se déroule de la manière suivante :

- L'identification et la description des cas d'utilisation. Cette étape va nous permettre de préciser les interactions des utilisateurs avec notre système.
- La modélisation des interactions entre agents par l'élaboration des diagrammes protocoles d'interaction.
- L'élaboration du diagramme de classes.

5.1. Identification et description des cas d'utilisation

Les cas d'utilisation sont la première étape d'analyse d'un système, ils ont pour but de recueillir, d'analyser et d'organiser les besoins et de récencer les grandes fonctionnalités d'un système. Ils décrivent les interactions du système avec l'utilisateur. Ils permettent de concevoir et construire un système adapté aux besoins de l'utilisateur.

Ils mettent en évidence les relations fonctionnelles entre les acteurs et le système étudié. Donc c'est une vue du système depuis son environnement extérieur.

5.1.1. Le diagramme de cas d'utilisation

La figure (3.6) illustre le diagramme de cas d'utilisation global de notre système. Dans ce diagramme nous avons trois acteurs principaux :

- **Le conducteur** : qui accède principalement à l'application pour chercher et réserver des places.
- **Le gestionnaire de parking** : qui utilise notre système pour créer un espace virtuel pour offrir des services de réservation des places de stationnement pour les conducteurs.
- **L'administrateur** : qui est le responsable de la gestion du système.

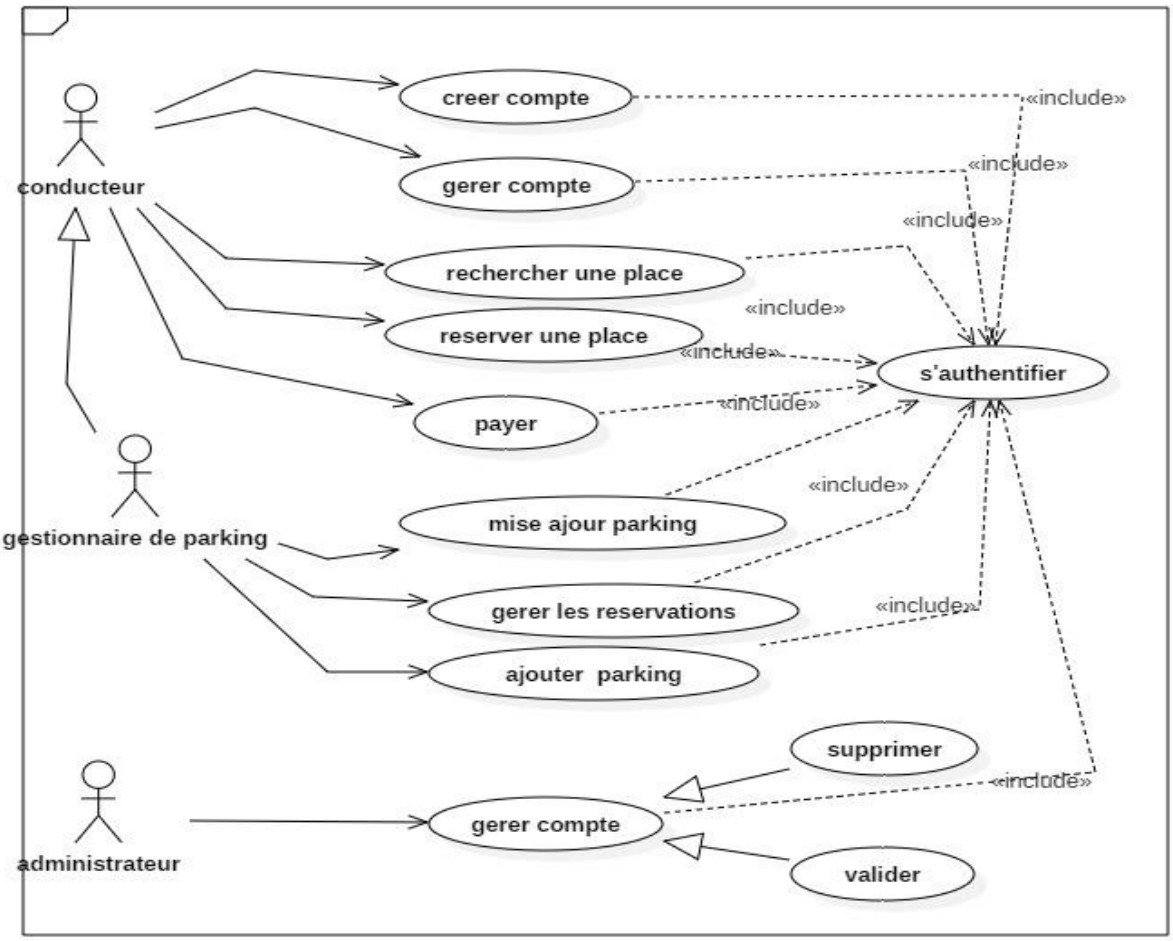


Figure 3.6: Diagramme de cas d'utilisation

5.1.2. Les diagrammes de séquences système

Dans la suite nous allons présenter quelques diagrammes de séquences décrivant les principaux cas d'utilisation de notre système.

- **Diagramme de séquence inscription**

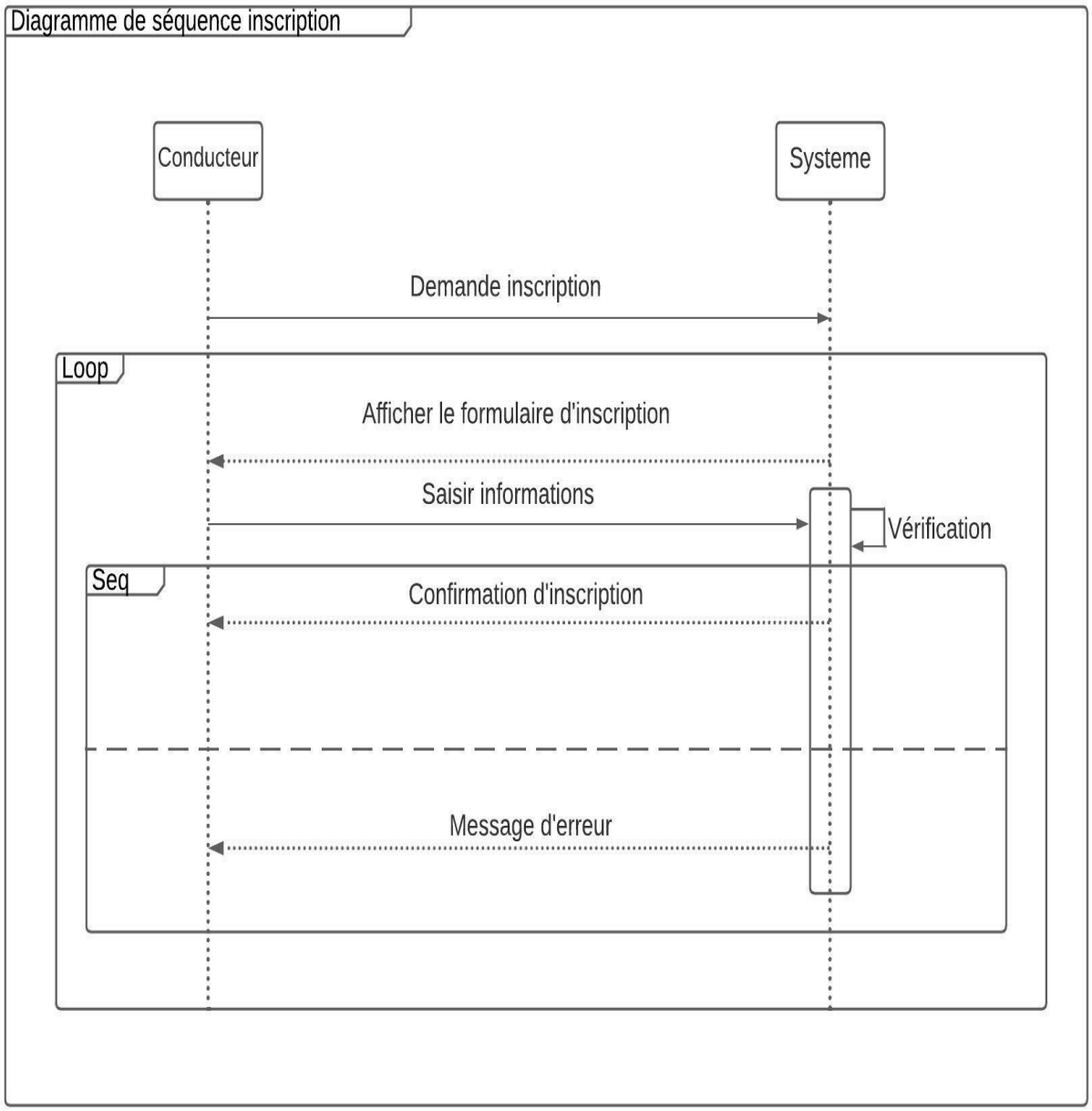


Figure 3.7: Diagramme de séquence inscription

- **Diagramme de séquence authentification**

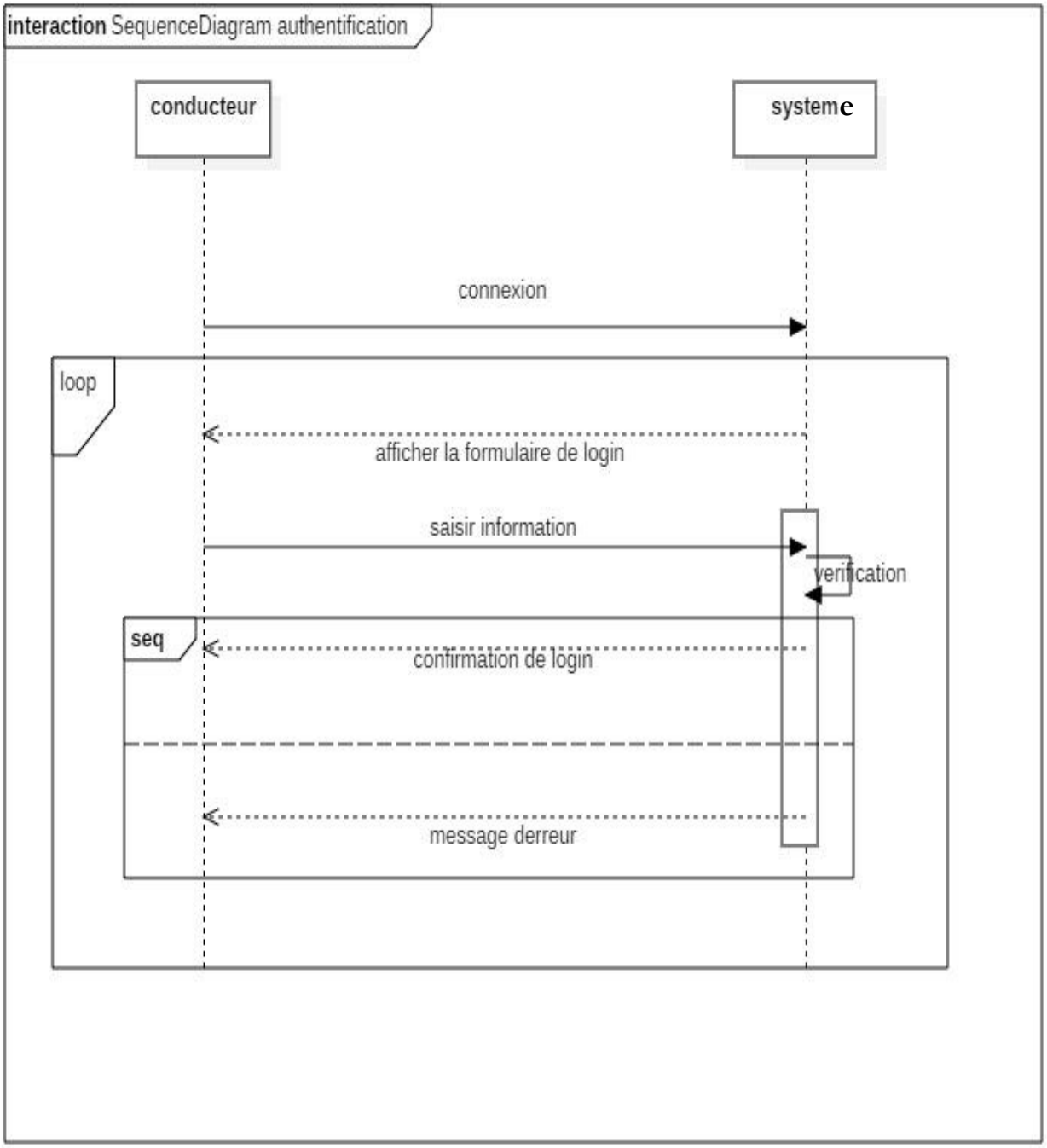


Figure 3.8: Diagramme de séquence authentification

- **Diagramme de séquence rechercher une place**

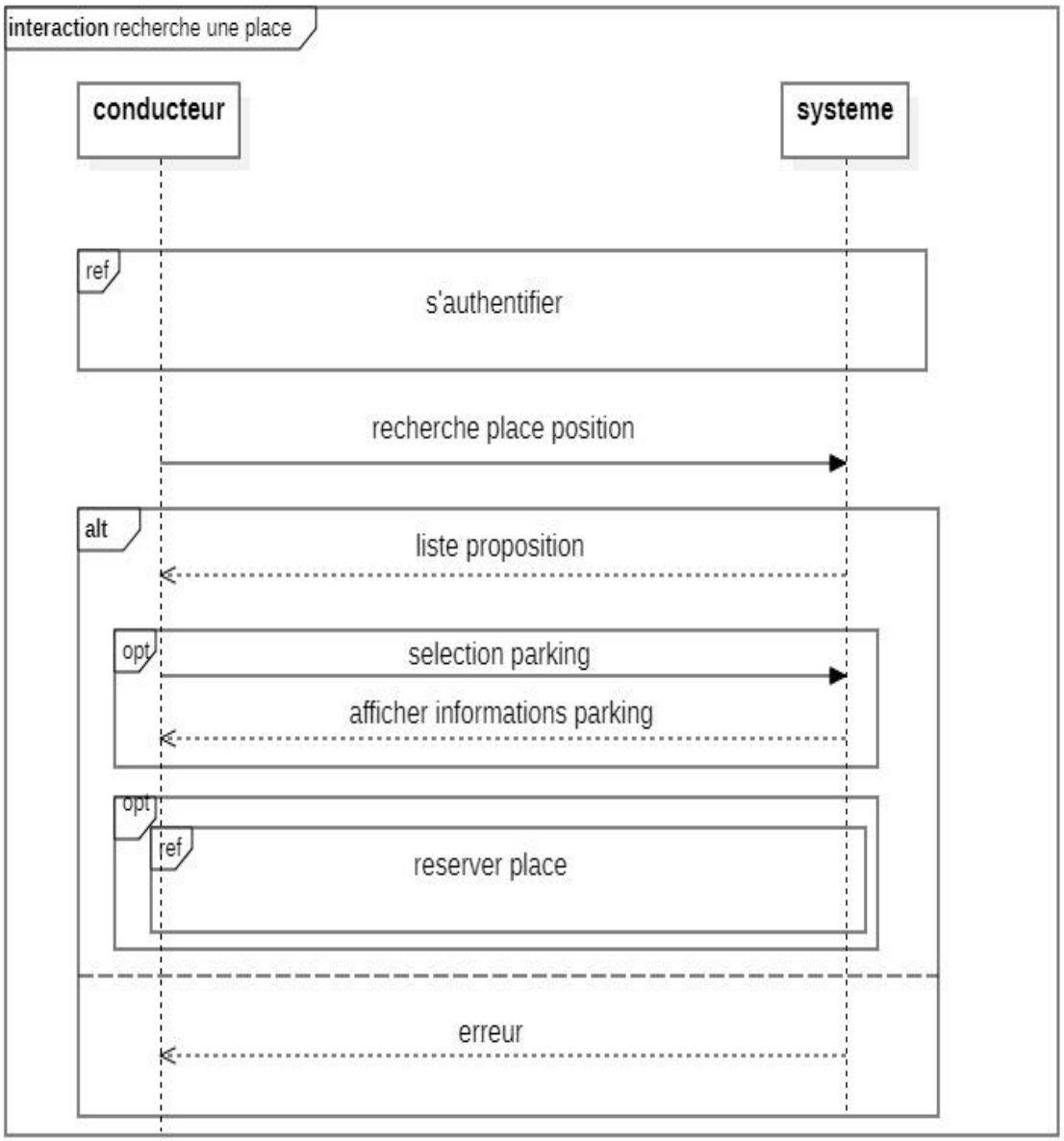


Figure 3.9: Diagramme de séquence rechercher une place

- **Diagramme de séquence réserver place**

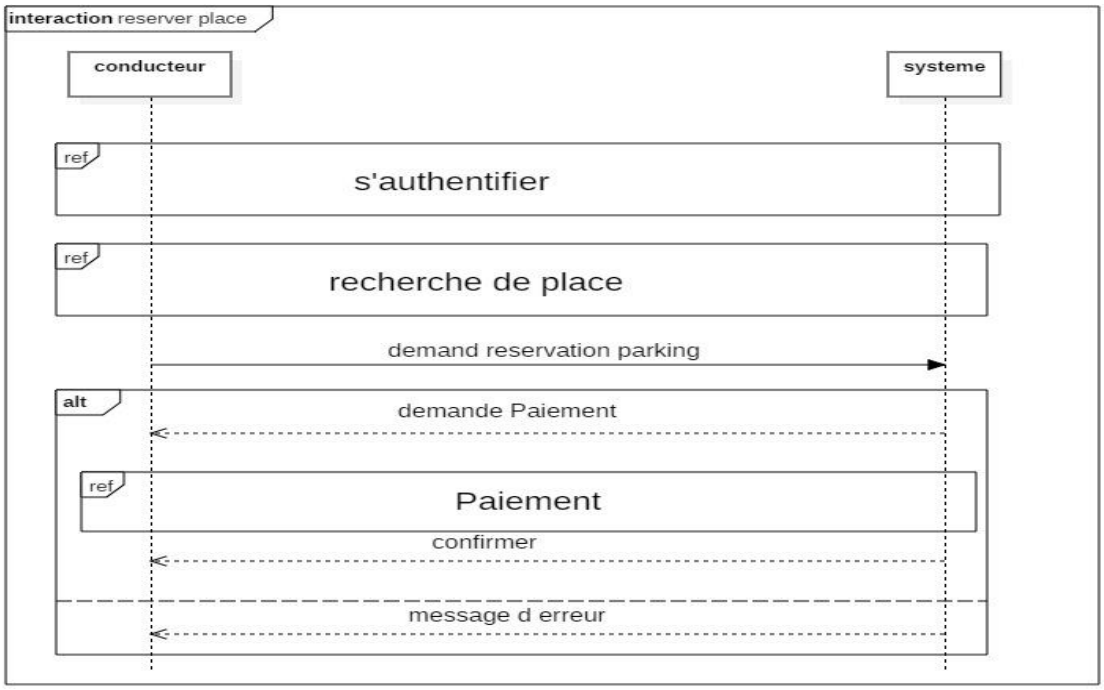


Figure 3.10: Diagramme de séquence réserver place

- **Diagramme de séquence ajouter parking**

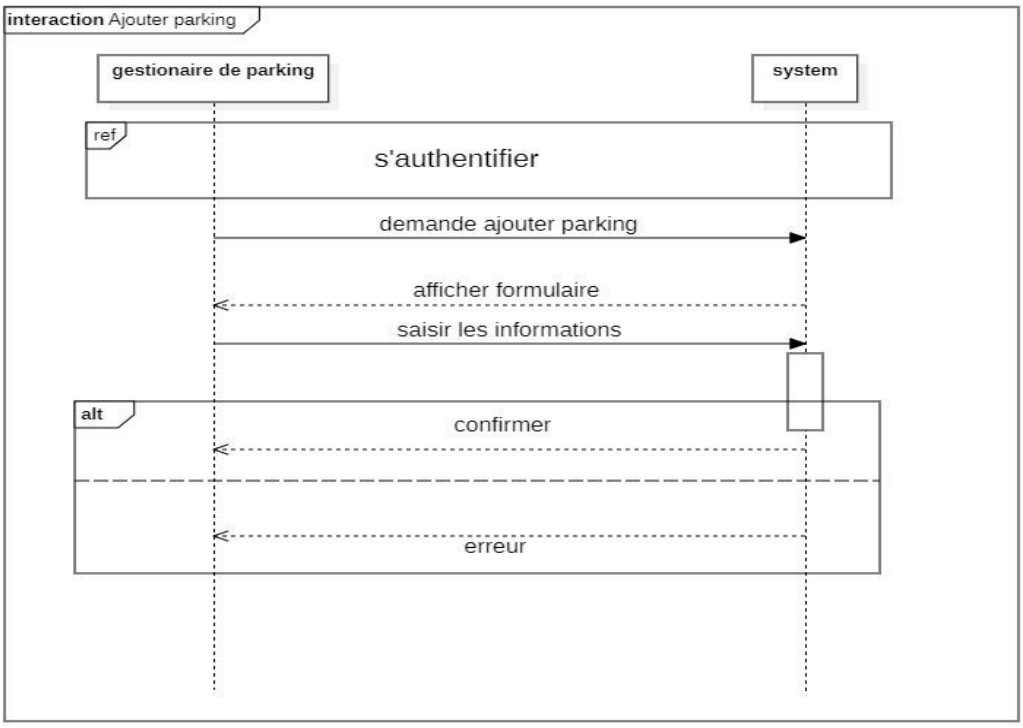


Figure 3.11: Diagramme de séquence ajouter parking

- **Diagramme de séquence mis à jour parking**

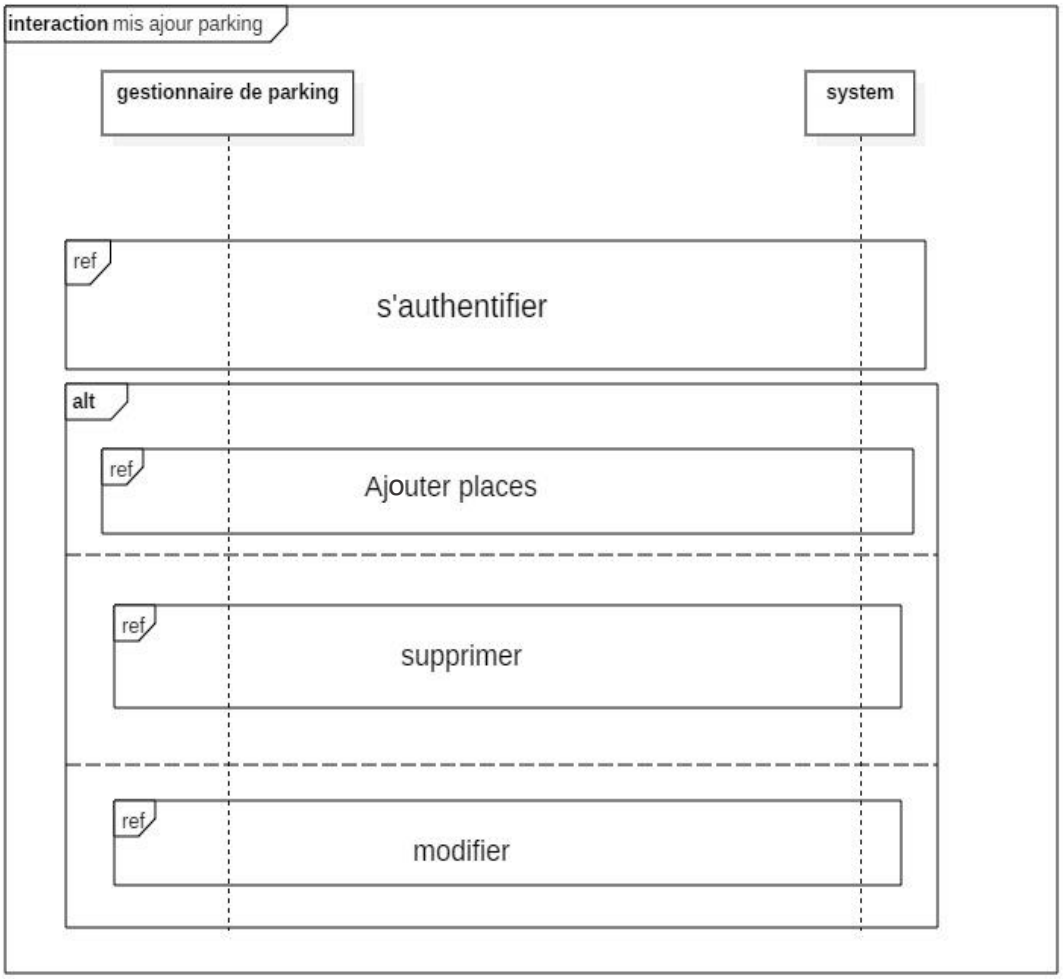


Figure 3.12: Diagramme de séquence mis à jour parking

5.2. Diagrammes de protocole d'interactions

5.2.1. Diagramme de protocole d'authentification

Dans ce diagramme l'agent conducteur envoie une requête d'authentification avec le login à l'agent Admin, Ce dernier vérifier les données, si les données sont valides alors il envoie le message (CONFIRM) et crée un agent réservation lié à ce conducteur, sinon il renvoie un message d'erreur (FAILLURE). La figure (3. 13) illustre le diagramme de protocole de l'authentification.

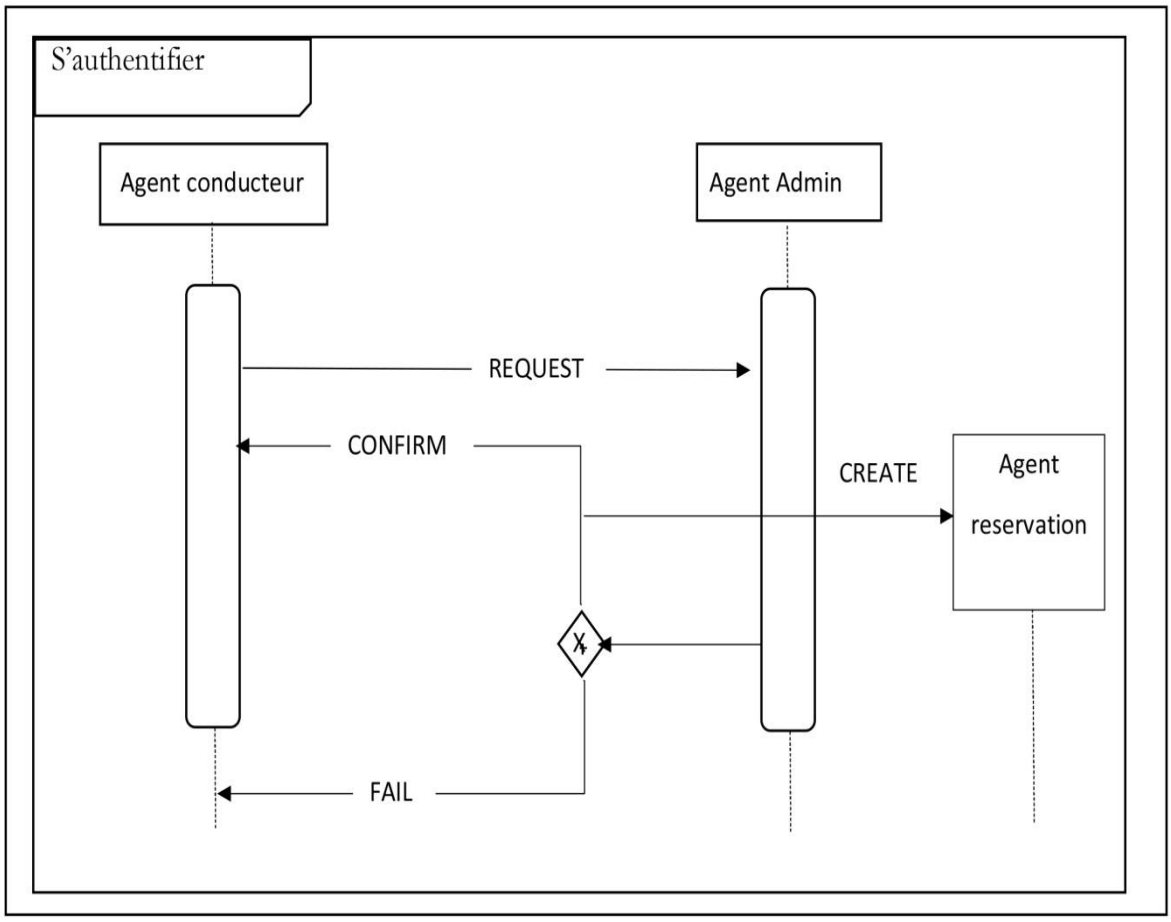


Figure 3.13: Diagramme de protocole d'authentification

5.2.2. Diagramme de protocole d'interactions rechercher une place

Dans ce diagramme l'Agent conducteur envoie une requête de recherche de places à l'agent réservation. L'agent réservation transmet cette requête aux différents agents parking dans le système, chaque Agent Parking traite la requête et répond par un message (INFORM) contenant la liste des places disponibles, lors de la réception de toutes les réponses et après les traitements nécessaires, l'agent réservation informe l'agent conducteur du résultat de recherche.

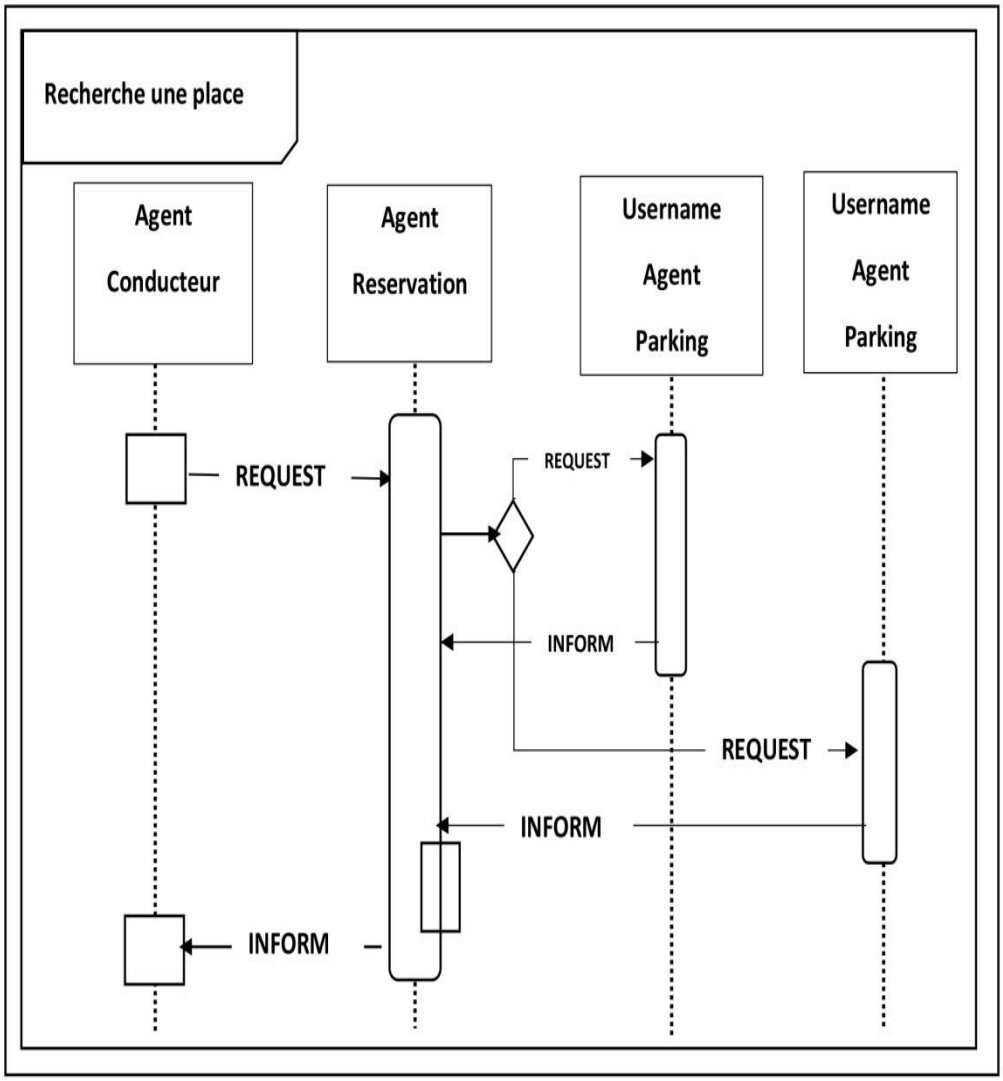


Figure 3.14: Diagramme de protocole d'interactions rechercher une place

5.2.3. Diagramme de protocole d'interactions réserver une place

L'agent Conducteur envoie une requête à l'agent réservation qui transmet cette requête aux agents parkings concernés par la demande de la réservation qui'ils sont repond par informe à l'agent reservation qui envoie une requete à l'agent paiment pour accomplir l'operation de paiment. Après le traitement de la requête de reservation l'agent reservation envoie un message de confirmation (CONFIRM) ou d'erreur (FAIL) au conducteur.

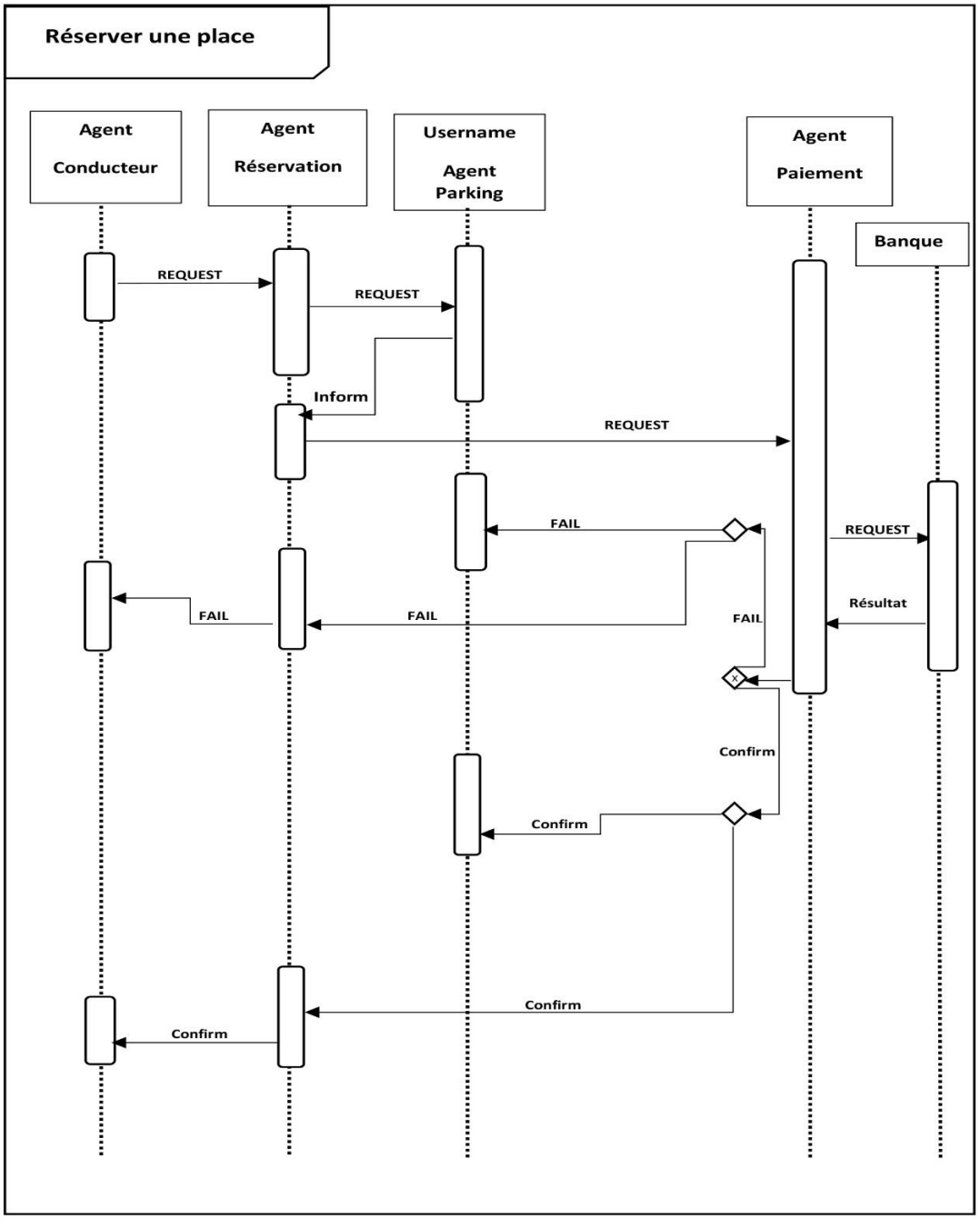


Figure 3.15: Diagramme de protocole d'interactions réserver une place

5.2.4. Diagramme de protocole d'interactions ajouter parking

Dans ce diagramme l'Agent parking envoie une requête (REQUEST) pour ajouter un parking à l'Agent Admin, si les informations sont correctes et complètes l'Agent Admin ajout le parking et répond par un message de confirmation (CONFIRM), sinon il répond par un message d'erreur (FAIL).

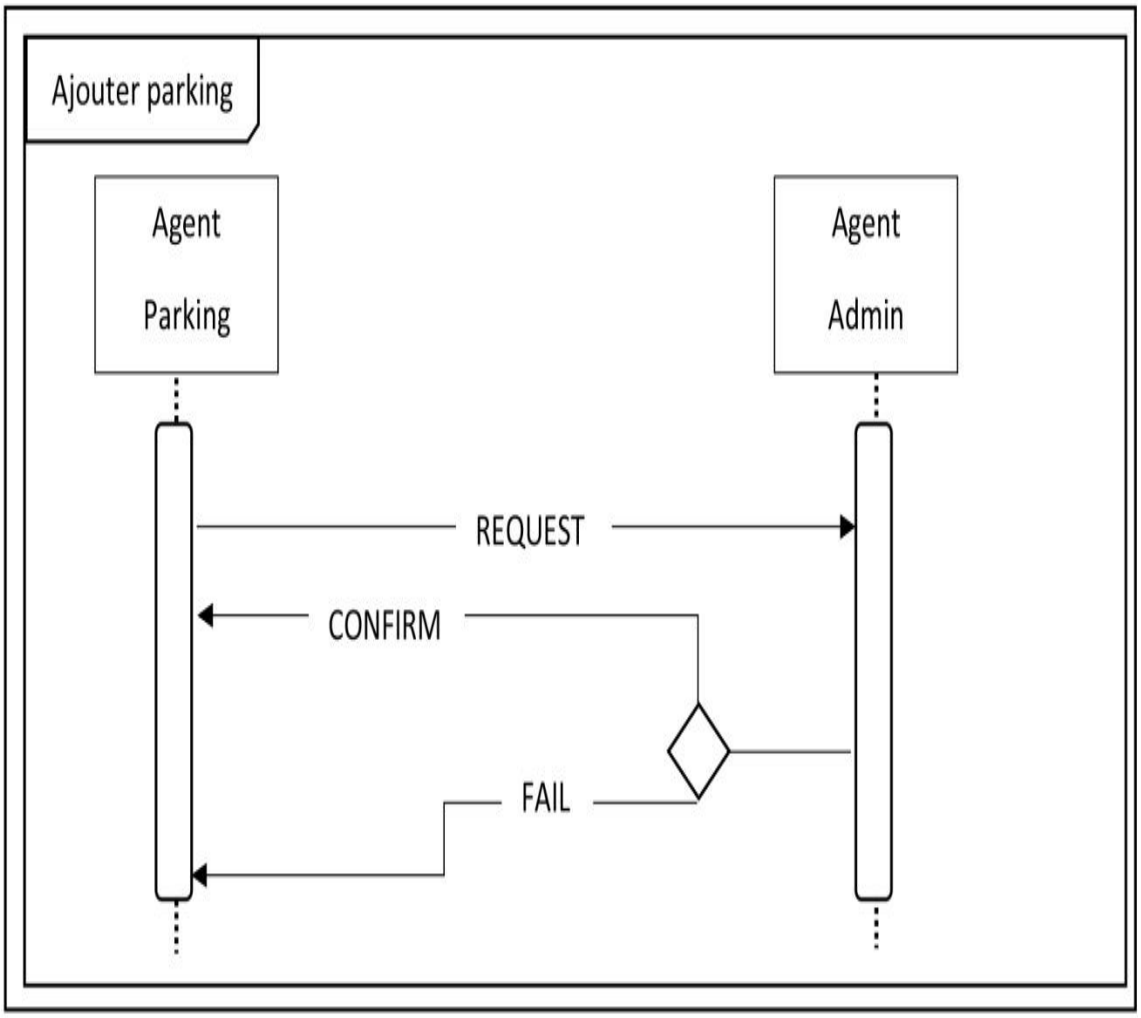


Figure 3.16: Diagramme de protocole d'interactions ajouter parking

5.3. Diagrammes de classes

Dans cette section nous passons à la conception de l'aspect statique de notre système. Nous commençons par l'élaboration du diagramme de classes représentant notre système. Ensuite nous allons présenter dans un diagramme séparé les différentes classes d'agent utilisé dans notre système.

5.3.1. Diagramme de classes du système

La figure (3.17) représente le diagramme de classes de notre système.

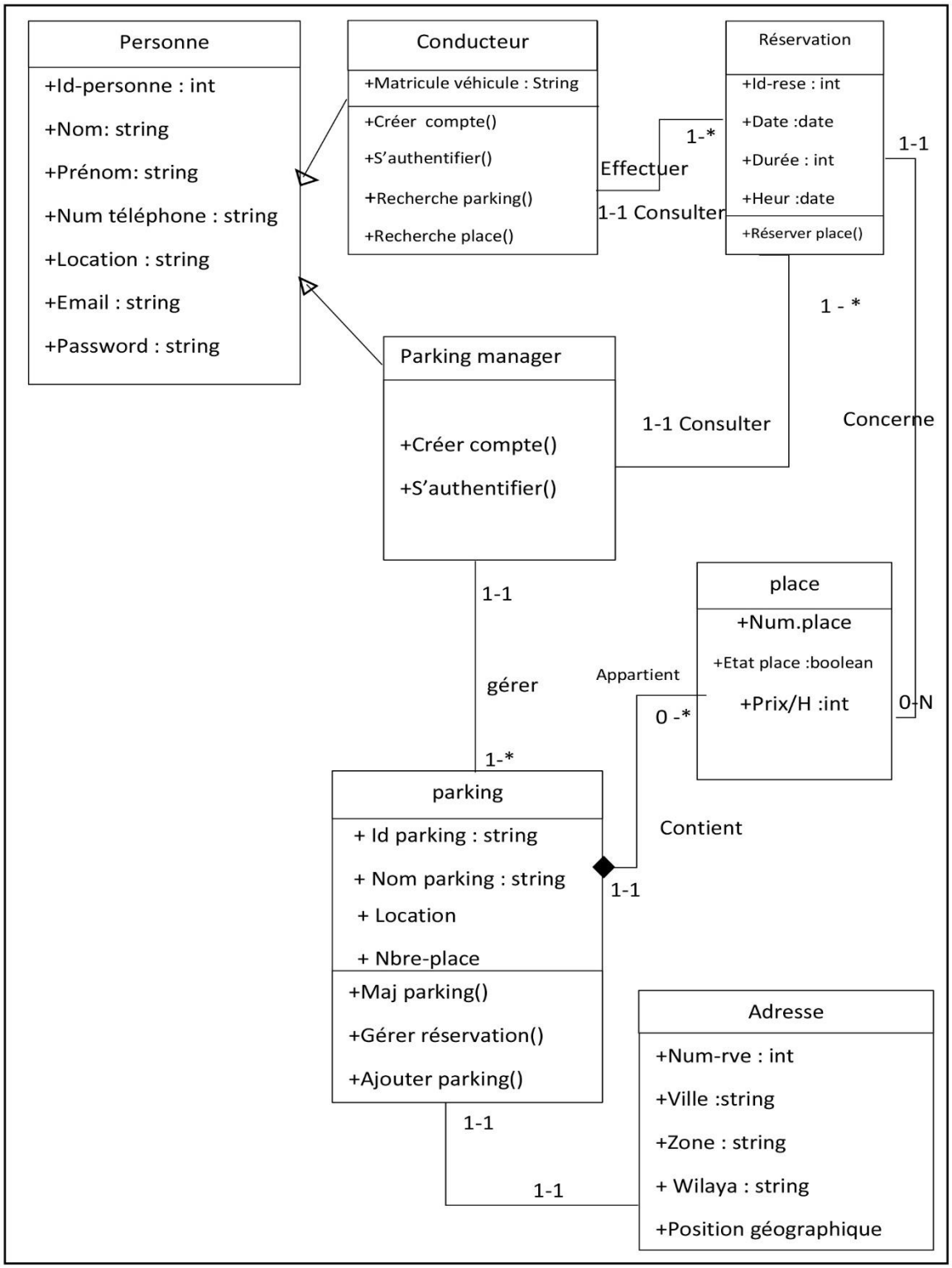


Figure 3.17: Diagramme de classes du système

5.3.2. Diagramme de classes agents

La figure (3.18) représente le diagramme de classes des agents de notre système.

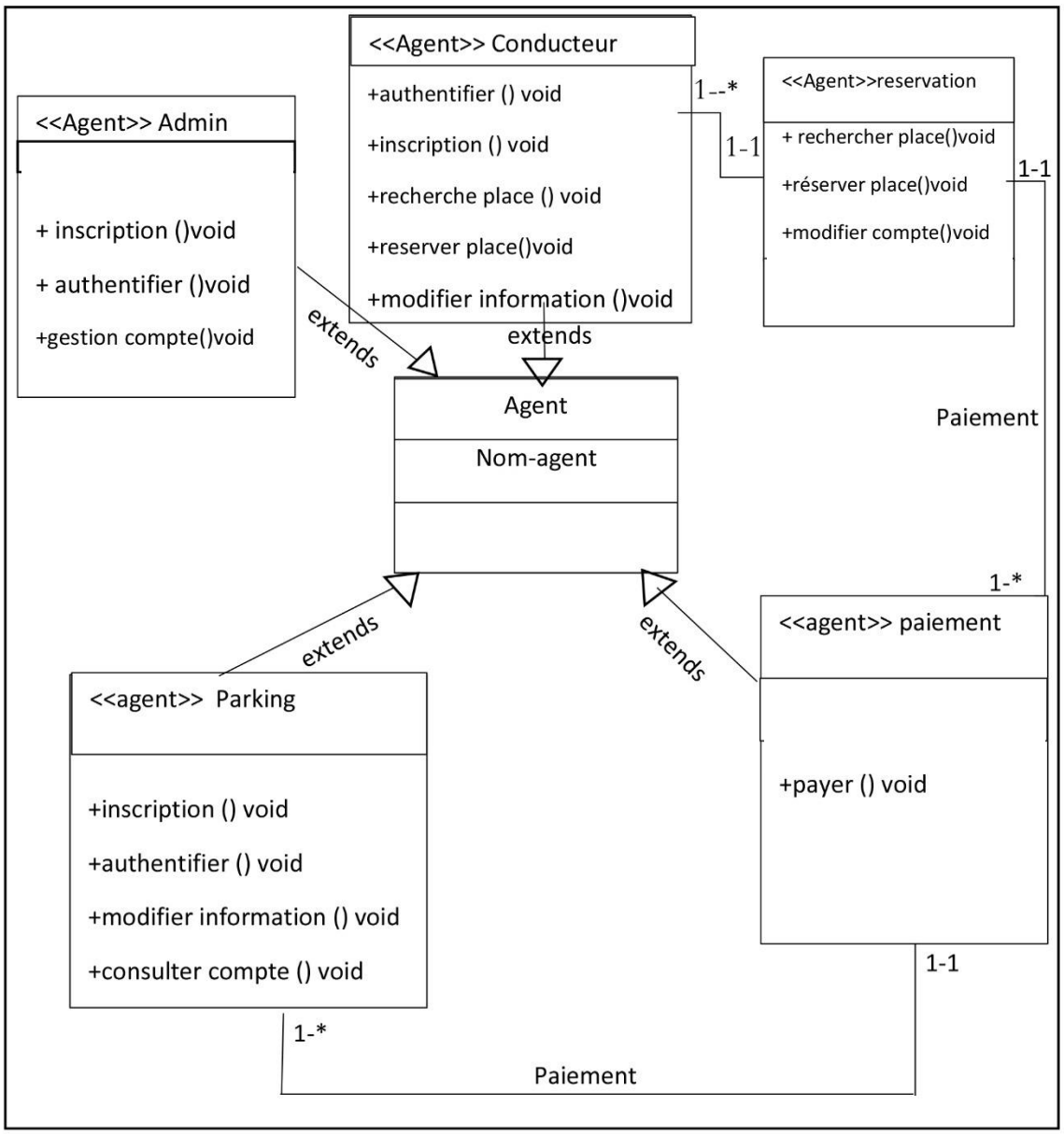


Figure 3.18: Diagramme de classes agents

6. Conclusion

Dans ce chapitre, nous avons présenté la phase d'analyse et de conception de notre système, en se basant sur la méthodologie voyelle et sur le langage AUML. Après l'identification de différentes composantes du système, nous avons élaboré les diagrammes AUML nécessaire pour sa modélisation, à savoir le diagramme de cas d'utilisation, le diagramme de protocole d'interaction et le diagramme de classes. Le chapitre cinq sera consacré à la réalisation et l'implémentation de ce système.

Chapitre4 : Réalisation du parking intelligent (Smart Parking)

Chapitre4 :

Réalisation du parking intelligent (Smart parking)

1. Introduction

Ce chapitre est consacré à la réalisation d'un systéme d'objets connectés pour le smart parking dans le but de contrôler les véhicules entrent dans un parking. Nous allons décrire l'objet connecté en présentant les paramètres de surveillance à utiliser, la liste des composants et les modules électroniques nécessaires à sa réalisation. Nous terminons par la présentation des étapes de mise en œuvre concrète de l'objet connecté, entre autres le branchement des capteurs et l'implémentation.

2. Analyse des besoins

Dans cette section, nous présentons également les besoins matériels et fonctionnels du Smart parking à réaliser, les choix hardwares et software effectués ainsi que le fonctionnement du système.

2.1. Description du système

L'objectif de ce travail est l'étude d'un système permettant de contrôler l'entrée d'un véhicule dans un parking par le contrôle des places libres et les réservations en ligne. Ce système est capable de trouver automatiquement les emplacements vides disponibles pour le stationnement. S'il y'a des places dans le **parking automatisé,** les nouveaux véhicules sont autorisés à entrer dans le parking, sinon l'entrée est bloquée en utilisant

la barrière Servomoteur. Les visiteurs peuvent voir l'état de la disponibilité des places à l'extérieur du parking sur un écran LCD 12c.

2.2. Besoins Matériels

Le tableau suivant illustre les différents composants électroniques nécessaires pour la réalisation de notre système.

Composant	**Quantité**
Carte Arduino UNO	1
IR proximity sensor	2
Servomoteur	1
Afficheur LCD I2c	1
Module Wifi	1
Condensateur	1
Jumpers	Plusieurs

Tableau 4.1: Composants électroniques utilisés dans le système de gestion de parking intelligent.

2.2.1. Carte Arduino UNO

Nous allons utiliser pour notre objet connecté la carte Arduino UNO Les motivations qui nous ont incités à choisir Arduino UNO sont :

- **Le prix (réduits)** : les cartes Arduino sont relativement peu coûteuses comparativement aux autres plates-formes. La version UNO est la moins chère des versions du module Arduino qui peut être assemblée à la main.
- **Multi plateforme** : le logiciel Arduino, écrit en C, tourne sous les systèmes d'exploitation Windows, Macintosh et Linux. Tandis que la plupart des systèmes à microcontrôleurs sont limités à Windows.
- **Logiciel Open Source et extensible** : Le logiciel Arduino et le langage C (pour la programmation de la carte) sont publiés sous licence open source.

- **Disponibilité** : les cartes Arduino sont disponibles dans le marché contrairement aux autres microcontrôleurs.

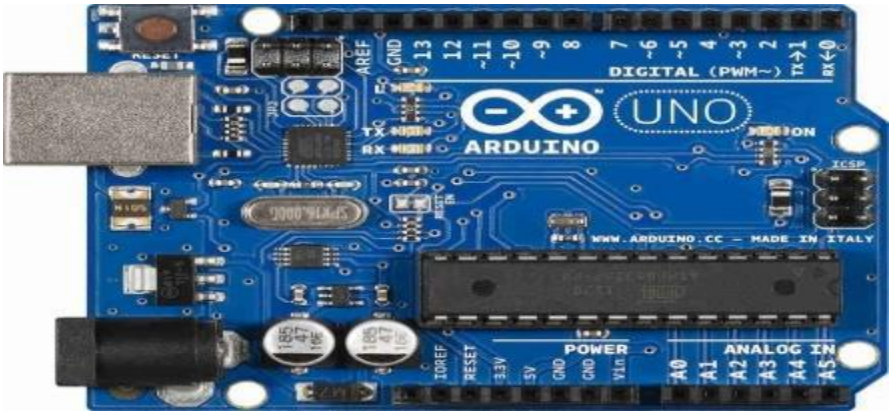


Figure 4.1: La carte Arduino UNO

2.2.2. IR proximity sensor

Les capteurs de détection de proximité mesurent l'énergie infrarouge (IR) réfléchie pour détecter la présence d'un objet ou d'une personne. Les appareils comprennent un pilote de LED intégré et, dans certains appareils, une LED intégrée. Les dispositifs de détection de proximité fournissent quatre courants de commande de LED programmables et des répétitions d'impulsions IR[40].

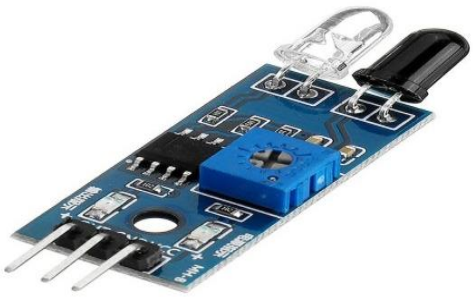


Figure 4.2: IR Proximity Sensor

2.2.3. LCD I2c

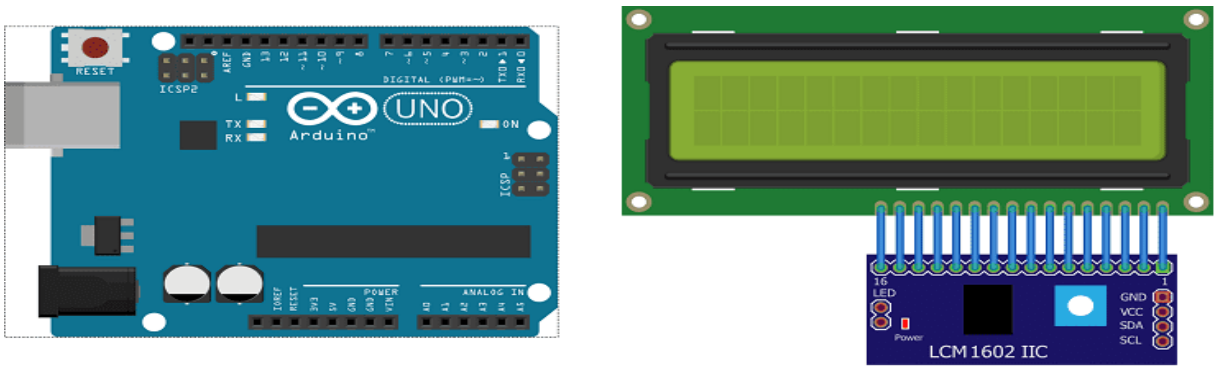
LCD I2C est un écran LCD normal. Il utilise juste un circuit à souder qui lui permet de fonctionner avec l'interface I2C. Ce qui rend le câblage plus facile.

Figure 4.3 : LCD I2c

A. Montage

Pour réaliser le montage il faut connecter :

- La broche SDA de module 12c de l'afficheur LCD au ANALOG PIN 4 de l'Arduino
- La broche SCL de module 12c de l'afficheur LCD au ANALOG PIN 5 de l'Arduino
- La broche GND de module 12c de l'afficheur LCD à la broche GND de l'Arduino
- La broche VCC de module 12c de l'afficheur LCD à la broche 5V de l'Arduino
- et après connecté les broches du module I2C avec les broches de l'écran LCD 16 × 2



Carte Arduino UNO **Afficheur LCD**

Figure 4.4 : Montage de LCD

2.2.4. Servomoteur

Un servomoteur est un moteur capable de maintenir une opposition à un effort statique et dont la position est vérifiée en continu et corrigée en fonction de la mesure.

C'est donc un système asservi. La position est définie avec une limite de débattement d'angle de 180 degrés, mais également disponible en rotation continue. Dans notre système un servomoteur de type microservo_9G a été utilisé pour l'ouverture et la fermeture de barrière d'entrée et de sortie du parking. La figure (4.5) présente le branchement d'un servomoteur avec la carte arduino.

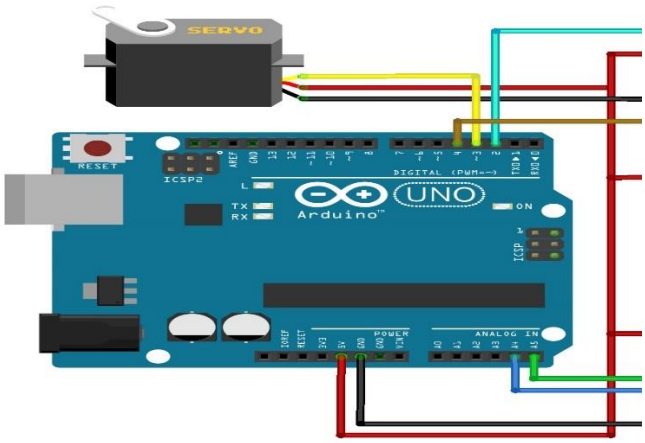


Figure 4.5 : Servomoteur

2.2.5. Module Wifi ESP8266

Le NodeMCU est un environnement de développement de logiciels open source et du matériel qui est construit autour d'un très bon marché System-on-Achip (SoC) appelé ESP8266. Le ESP8266, conçu et fabriqué par Espressif Systems, contient tous les éléments cruciaux de l'ordinateur moderne : CPU, RAM, réseau (sans fil), et même un système d'exploitation moderne et SDK. NodeMCU dans ce prototype utilisé un module de microcontrôleur avec un système Wi-Fi qui a été programmé en utilisant IDE Arduino. La version de NodeMCU utilisait NodeMCU.

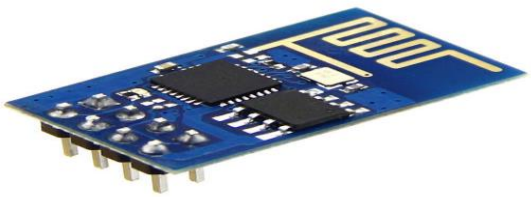


Figure 4.6: Module Wifi ESP8266

2.2.6. Condensateur :

Un condensateur est un composant électronique ou électrique dont l'intérêt de base est de pouvoir recevoir et rendre une charge électrique, dont la valeur est proportionnelle à la tension. Il se caractérise par sa Capacité électrique. Son comportement électrique idéal.

Il est utilisé principalement pour :

- Stabiliser une alimentation électrique (il se décharge lors des chutes de tension et se charge lors des pics de tension) ;
- traiter des signaux périodiques;
- Stocker de l'énergie, auquel cas on parle de supercondensateur [41].

Figure 4.7: Condensateur

2.2.7. Jumpers

Les Jumpers (fils de connexion) sont utilisés pour établir des connexions entre les différents éléments et les broches de la carte Arduino.

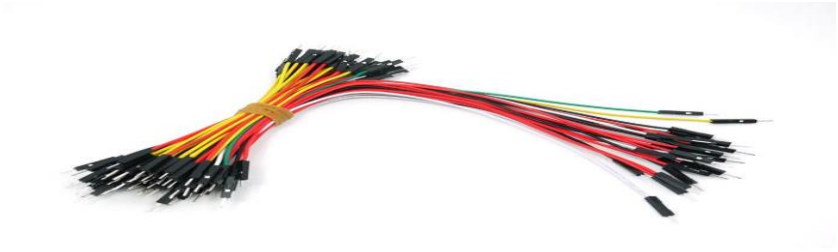


Figure 4.8: Jumpers

2.3. Besoins fonctionnels

Les besoins fonctionnels sont résumés par :

- Le prélèvement des données (état de place de parking).
- L'interaction et la communication avec l'application SMA pour vérifier les réservations des places et mettre à jour les etats des places.

- La gestion et le contrôle du parking (l'entrée et la sortie des vehicules).

3. Langages et outils utilisés

3.1. L'IDE Arduino

Nous avons utilisé la plateforme de développement Arduino, compatible avec le type du microcontrôleur utilisé.

3.2. L'IDE pour la programation en Java

Ce qui concerne la couche communication, nous avons l'implémenter en utilisant conjointement l'environnement Arduino et Eclipse sous JAVA.

3.3 TINKERCAD

Le logiciel en ligne Circuit TINKERCAD (Figure 4.9) permet de réaliser des circuits avec des composants (capteurs, servo-moteurs, voltmètres, …) et d'en simuler le fonctionnement piloter par un programme en lignes de commande Arduino.

Il permet, après un enregistrement obligatoire avec une adresse mail, de partager les conceptions, donc de profiter de milliers d'exemples [42].

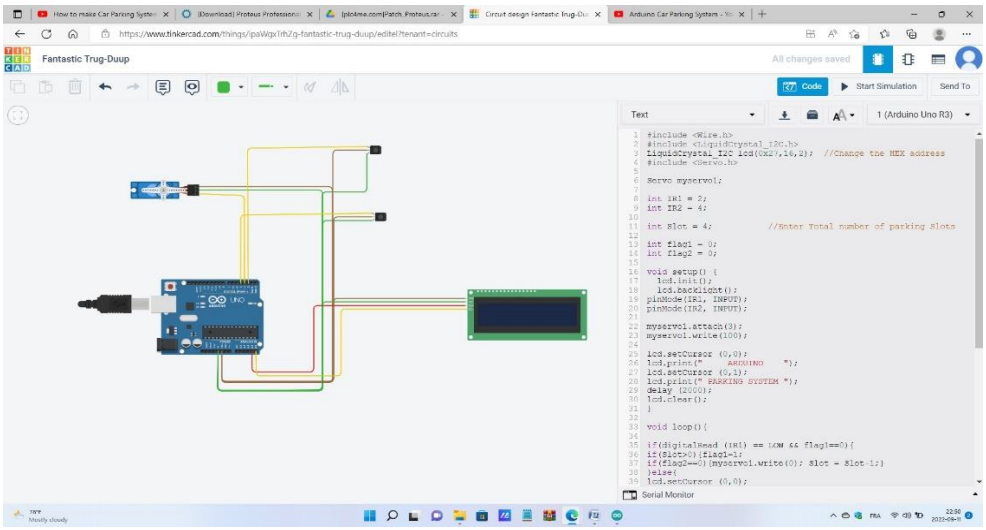


Figure 4.9: Simulation de système sur TINKERCARD

4. Réalisation du système

4.1. Architecture matérielle du système

La figure (4.10) represente l'architecture materielle de notre système de stationnement intelligent composé des elements suivants :

- Arduino UNO
- Deux capteurs IR
- Servomoteur
- LCD 16×2 et module I2C
- Module Wifi ESP8266
- Câble USB pour télécharger le code

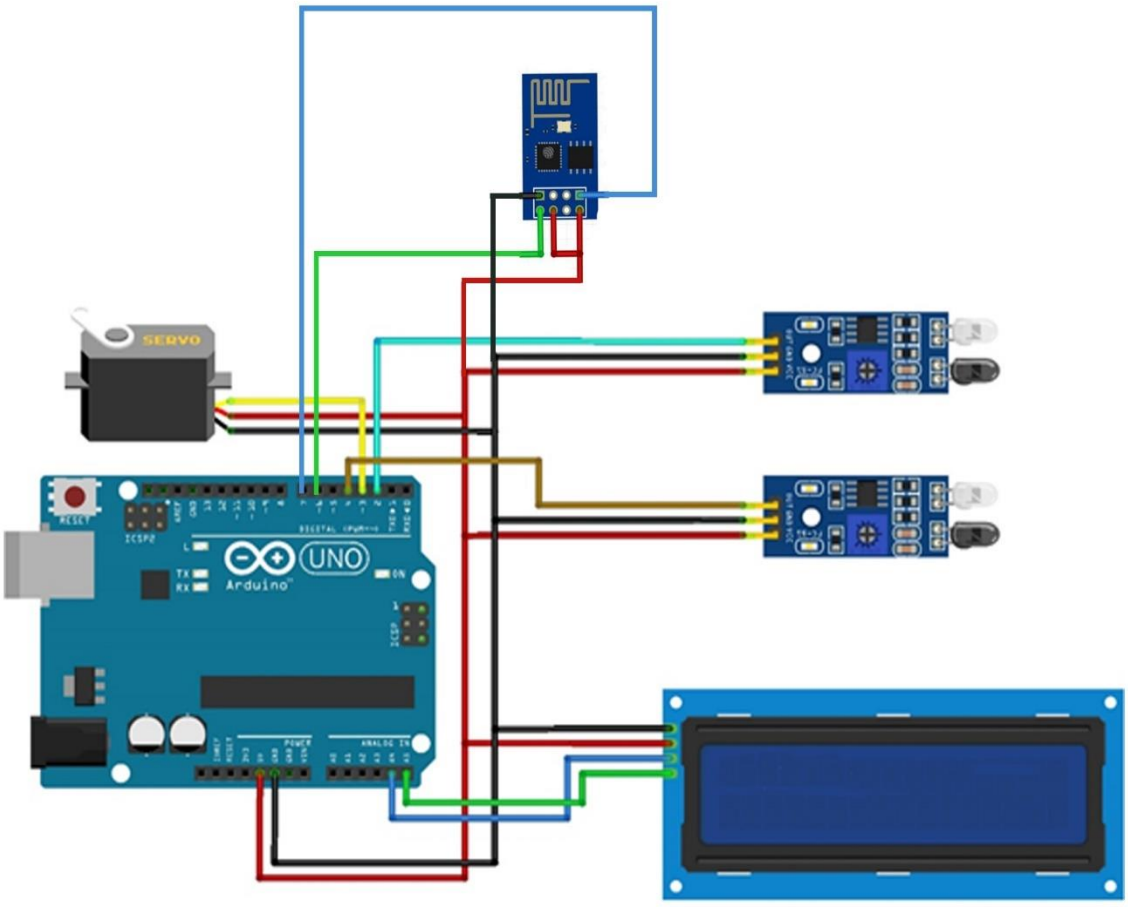


Figure 4.10 : Schéma de circuit de stationnement automatique

4.2. Le système réalisé

La figure (4.11) représente le système realisé.

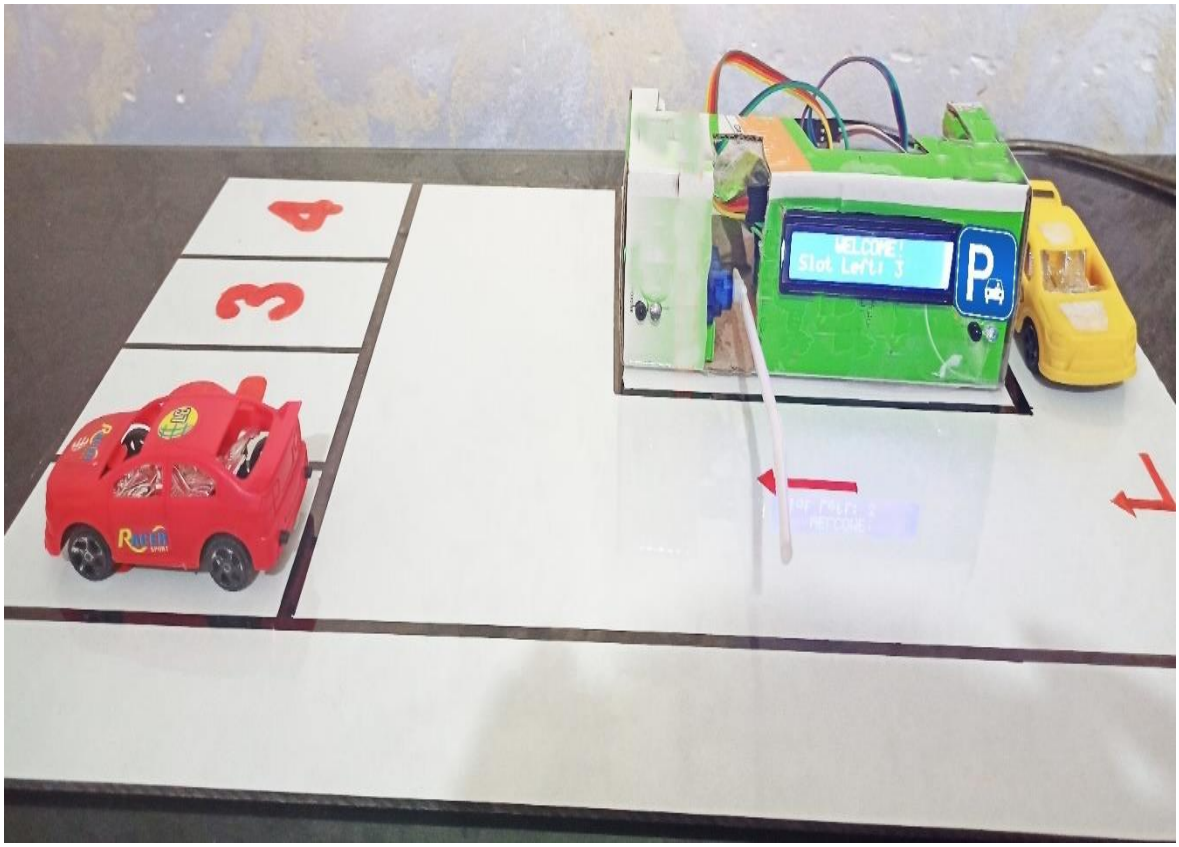


Figure 4.11 : Stationnement intelligent

4.3. Principe de fonctionnement

Le diagramme d'activité sous-dessous montre le principe de fonctionnement de notre système qu'on peut le résumer par les points suivants :

- Détecter la présence d'un véhicule en entrée ou en sortie du parking.
- Vérification s'il y a une réservation pour le véhicule à entrer.
- Vérifier le nombre de place vide.
- Ouvrir la barrière tout en gérant l'entrée et la sortie des voitures.

4.3.1. Diagramme d'activité

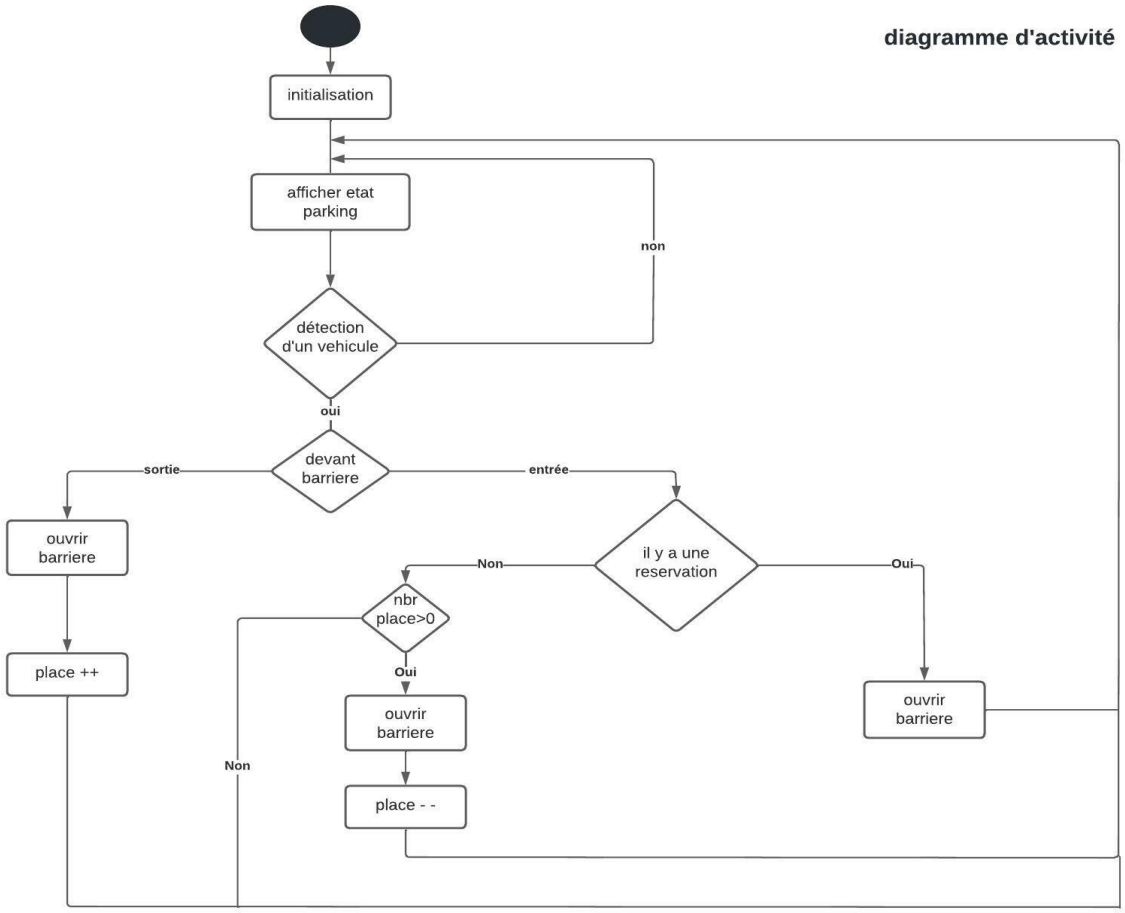


Figure 4.12: Diagramme d'activité

4.3.2. Description des codes

Le code doit être charger sur arduino et installé avec les bibliothèques <LiquidCrystal_I2C.h> et <Wire.h>pour que les éléments connectés fonctionnent bien.

```
// I2C library et servomotor library
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27,16,2);
#include <Servo.h> //library pour servo motor

//declaration des variables
Servo myservo1;
int IR1 = 4; // IR Sensor 1
int IR2 = 7; // IR Sensor 2
int Slot = 4; //Enter Total number of parking Slots
int flag1 = 0;
int flag2 = 0;
```

```
//initialisation
void setup()
{
lcd.init();
lcd.backlight();
pinMode(IR1, INPUT);
pinMode(IR2, INPUT);
myservo1.attach(9);
myservo1.write(100);
lcd.setCursor (0,0);
lcd.print("   ARDUINO  ");
lcd.setCursor (0,1);
lcd.print(" PARKING SYSTEM ");
delay (2000);
lcd.clear();
}
```

```
//le processus
void loop(){
if(digitalRead (IR1) == LOW && flag1==0){
if(Slot>0){flag1=1;
if(flag2==0){myservo1.write(0); Slot = Slot-1;}
}else{
lcd.setCursor (0,0);
lcd.print("  SORRY   ");
lcd.setCursor (0,1);
lcd.print(" Parking Full ");
delay (3000);
lcd.clear();
}
}
if(digitalRead (IR2) == LOW && flag2==0){flag2=1;
if(flag1==0){myservo1.write(0); Slot = Slot+1;}
}
if(flag1==1 && flag2==1){
delay (1000);
myservo1.write(100);
flag1=0, flag2=0;
}
lcd.setCursor (0,0);
lcd.print("  WELCOME!  ");
lcd.setCursor (0,1);
lcd.print("Slot Left: ");
lcd.print(Slot);
}
```

5. Conclusion

Dans ce chapitre nous avons procédé par la réalisation d'un système d'objets connectés pour la surveillance de parking. Nous avons expliqué, en détail, les étapes à suivre pour réaliser ce système avec le matériel utilisé, le chapitre suivant est consacré l'implémentation de l'application SMA et la réalisation de système complet.

Chapitre 5 : réalisation et implimentation

Chapitre 5
Réalisation et implémentation

1. Introduction

Dans le chapitre trois de ce mémoire, nous avons proposé une architecture basée agent pour un système de smart parking. Dans ce chapitre, nous allons passer à la dernière étape qui est la réalisation du système complet. Cette étape est cruciale pour la mise en place de ce que nous avons fait auparavant. Nous entamerons ce chapitre par la présentation des différents langage et outils utilisés pour développer notre système. Ensuite, nous allons expliquer brièvement la manière d'implémentation de notre système avec Jade. Nous terminerons ce chapitre par la présentation de quelques interfaces et fonctionnalité de notre application et de différents agents du système en utilisant les outils de Jade.

2. Langages et outils de développement

Nous présentons dans cette section les différents langages et outils utilisés pour le développement de notre application.

2.1. Langage Java

La technologie Java définit à la fois un langage de programmation orienté objet et une plateforme informatique. Crée par l'entreprise Sun Microsystems (souvent juste appelée "Sun") en 1995, et reprise depuis par la société Oracle en 2009, la technologie Java est indissociable du domaine de l'informatique et du Web. On la retrouve donc sur les ordinateurs, mais aussi sur les téléphones mobiles, les consoles de jeux, etc. L'avènement du Smartphone et la puissance croissante des ordinateurs, ont entraîné un regain d'intérêt pour ce langage de programmation [43].

2.2. Plateforme Jade

JADE (Java Agent Development Framework) est une plate-forme multi-agents développée en Java par CSELT (Groupe de recherche de Gruppo Telecom, Italie) qui a comme but la construction des systèmes multi-agents et la réalisation d'applications conformes à la norme FIPA (FIPA, 1997). JADE comprend deux composantes de base : une plate-forme agents compatible FIPA et un paquet logiciel pour le développement des agents Java [44].

2.3. Composants de Jade

JADE est un middleware qui facilite le développement de systèmes multi-agents. Il comprend [45]:

- Un environnement d'exécution (**Runtimeenvironment)** où les agents JADE peuvent "vivre" et qui doit être actif sur un hôte donné avant, un ou plusieurs agents peuvent être exécutés sur cet hôte.
- Une bibliothèque de classes (**Library)** que les programmeurs peuvent l'utiliser (directement ou en se spécialisant) pour développer leurs mandataires.
- Une suite d'outils graphiques (**Graphicaltools**) qui permet d'administrer et de surveiller l'activité des agents en cours d'exécution.

2.4. JADE-LEAP

JADE-LEAP est une version modifiée de la plate-forme JADE qui peut s'exécuter non seulement sur des PC et des serveurs, mais également sur des appareils à ressources limitées tels que les téléphones mobiles, à condition qu'ils soient compatibles Java (MIDP 1.0 est l'exigence minimale). Il est important de noter que JADE-LEAP fournit les mêmes API par rapport à JADE afin qu'un agent développé pour s'exécuter sur JADE puisse s'exécuter sur JADE-LEAP sans AUCUNE modification. Les différences entre JADE et JADE-LEAP se limitent aux mécanismes de communication de bas niveau invisibles pour les applications et les utilisateurs [46].

2.5. Android studio

Android Studio [47] est l'environnement de développement que Google propose à ses développeurs pour créer des applications Android.Présenté à la Google I/O en 2013, Android Studio propose un environnement de travail complet pour développer des applications sous Android. Vous pouvez y trouver un assistant permettant de créer rapidement un projet pour tous les périphériques Android, smartphones et tablettes, mais aussi Android Wear, Android auto, Google Glass et Android TV. L'assistant va se charger de récupérer le bon SDK Android, de paramétrer l'environnement de développement et de créer un émulateur pour tester votre appli pour la plateforme choisie.

L'éditeur de code d'Android Studio intègre des fonctions intelligente comme l'auto complétion ou l'analyse du code et met à disposition un outil permettant de gérer les différentes traductions de votre application. L'IDE de Google permet aussi de visualiser le résultat de votre travail sur différentes tailles d'écran et différents appareils émulés. Un moniteur permet aussi de vérifier l'optimisation de votre application en affichant l'utilisation de la mémoire durant son fonctionnement.

2.6. FIREBASE

FIREBASE [48] est une base de données en temps réel qui permet de stocker des arborescences de listes d'objets. Il permet de synchroniser les données entre différents appareils. C'est une base de données NoSQL JSON permettant de stocker, de synchroniser et d'interroger les données des applications à l'échelle mondiale.

2.7. JDBC

La technologie JDBC (Java DataBaseConnectivity) est une API fournie avec Java (depuis sa version 1.1) permettant de se connecter à des bases de données, c'est-à-dire que JDBC constitue un ensemble de classes permettant de développer des applications capables de se connecter à des systèmes de gestion de bases de données (SGBD).

L'API JDBC a été développée de telle façon à permettre à un programme de se connecter à n'importe quelle base de données en utilisant la même syntaxe, c'est-à-dire

que l'API JDBC est indépendante du SGBD de plus, JDBC bénéfice des avantages de Java, dont la portabilité du code, ce qui lui vaut en plus d'être indépendant de la base de données d'être indépendant de la plate-forme sur laquelle elle s'exécute [50].

2.8. L'IDE Eclipse

L'IDE Eclipse (Integrated Développent Environnent) est célèbre pour notre environnement de développement intégré Java (IDE), mais nous avons un certain nombre d'IDE plutôt sympas, y compris notre IDE C / C ++, JavaScript / TypeScript IDE, PHP IDE, etc.

Vous pouvez facilement combiner la prise en charge de plusieurs langues et d'autres fonctionnalités dans l'un de nos packages par défaut, et Eclipse Marketplace permet une personnalisation et une extension pratiquement illimitées [51].

3. Implémentation des agents avec Jade

JADE est un middleware qui facilite le développement des systèmes multi agents (SMA). Dans Jade, les agents sont créés dans des conteneurs où chaque conteneur peut contenir plusieurs agents. L'ensemble de conteneurs constituent une plateforme. Chaque plateforme doit contenir au moins le conteneur principalqui est appelé main-container. Ce dernier qui possède trois modules principaux:

- **DF** « Directory Facilitator » fournit un service de « pages jaunes » à la plate-forme
- **ACC** « Agent Communication Channel » gère la communication entre les agents
- **AMS** « Agent Management System » supervise l'enregistrement des agents, leur authentification, leur accès et l'utilisation du système.

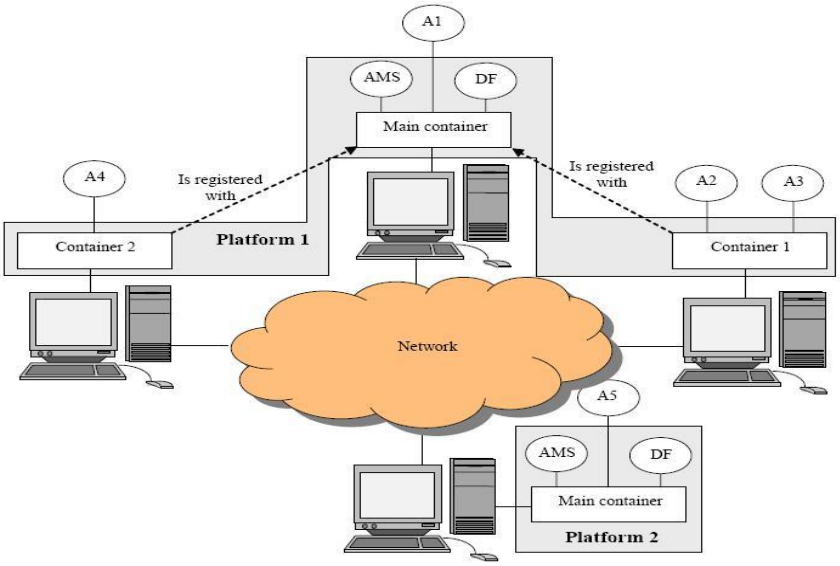


Figure 5.1: Jade Containers and Platform

Nous allons présenter dans la suite le lancement des différents conteneurs et la manière de création des agents de notre système.

3.1. Lancement des conteneurs

Chaque instance en cours d'exécution de l'environnement d'exécution JADE est appelée un conteneur car il peut contenir plusieurs agents. L'ensemble des conteneurs actifs est appelé une plate-forme. Un seul conteneur principal spécial doit toujours être actif sur une plateforme et tous les autres conteneurs sont créés dès qu'ils démarrent. Il s'ensuit que le premier conteneur pour démarrer sur une plate-forme doit être un conteneur principal tandis que tous les autres conteneurs doivent être "normaux" (c'est-à-dire conteneurs non principaux) et doivent « être informés » où trouver (hôte et port) leur conteneur principal (c'est-à-dire le principal conteneur pour s'enregistrer).

3.2. Lancement du conteneur principal (main container)

La figure (**5.2)** exprime la classe principale de notre application qui consiste au lancement du conteneur principale avec son GUI graphique par un code JAVA.

```java
package sma;
import jade.core.Runtime;

import jade.core.Profile;
import jade.core.ProfileImpl;
import jade.util.ExtendedProperties;
import jade.wrapper.AgentContainer;
import jade.util.leap.Properties;
import jade.wrapper.ControllerException;
public class maincontainer {
public static void main(String[]args) {

try {
    Runtime runtime=Runtime.instance();
    Properties properties=new ExtendedProperties();
    //ProfileImpl profileImpl=new ProfileImpl(properties);
    properties.setProperty(Profile.GUI, "true");
    Profile profile=new ProfileImpl(properties);
    AgentContainer mainContainer=runtime.createMainContainer(profile);
    mainContainer.start();
} catch (ControllerException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
}

}
```

Figure 5.2: Lancement du conteneur principal (main container)

3.3. Lancement du conteneur simple (simple container)

La figure (5.3) exprime la classe SimpleContainer comme exemple du lancement d'un conteneur secondaire avec Jade.

```java
package sma;

import jade.core.Runtime;

import jade.core.AID;

import jade.core.Profile;
import jade.core.ProfileImpl;
import jade.wrapper.AgentContainer;
import jade.wrapper.ControllerException;
import jade.wrapper.AgentController;
public class SimpleContainer {
public static void main(String[]args) {


    try {
        Runtime runtime=Runtime.instance();
        Profile profile=new ProfileImpl(false);
        profile.setParameter(Profile.MAIN_HOST, "localhost");
        AgentContainer agentContainer=runtime.createAgentContainer(profile);

        agentContainer.start();
    } catch (ControllerException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }

}
```

Figure 5.3 : Lancement du conteneur Simple Container

3.4. Code pour la création des agents

Notre système multi-agents contient :

- Des agents qui cherchent une place de stationnement (conducteurs).
- Des agents qui proposent leurs services parkings.
- L'agent de réservation qui devrait offrir la meilleure proposition de parking à l'agent conducteur.

Créer un agent JADE est aussi simple que de définir une classe qui étend la classe jade.core.Agent et implémenter la méthode setup().La figure (5 .4)représente un exemple des agents de notre système, l'agent conducteur.

```
public class ConducteurAgent extends GuiAgent {

transient protected Espaceconducteur Interface_sa;
private AID[] listparking ;
public ConducteurAgent() {
    this.Interface_sa = new Espaceconducteur(this);
  }
    private static final long serialVersionUID = -3076700735500511297L;



    @Override
    public void setup() {
        Interface_sa = new Espaceconducteur(this);
        Interface_sa.setVisible(true);


        System.out.println("Initialisation de l'agent "+this.getAID().getName());

        addBehaviour(new TickerBehaviour(this, 10000) {

            @Override
            protected void onTick() {

                try {
                    DFAgentDescription description= new DFAgentDescription();
                    ServiceDescription serviceDescription=new ServiceDescription();
```

Figure 5.4 : Exemple sur la création des agents (Agent conducteur)

Le comportement qu'un agent doit faire est représenté comme des comportements (behaviours) de JADE. Les comportements des agents sont ajoutés en utilisant la méthode addBehaviour. La figure (5.5) présente un exemple sur l'implémentation des comportements de l'agent conducteur.

```
addBehaviour(new OneShotBehaviour() {

    @Override
    public void action() {
        // TODO Auto-generated method stub
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            try {
                Class.forName("com.mysql.cj.jdbc.Driver");
            Connection con=DriverManager.getConnection("jdbc:mysql://localh

            String query1="insert into paiment(Nom_conducteur,nom_parking,

            PreparedStatement pst1= con.prepareStatement(query1);
            pst1.setString(1, Interface_sa.nomconducteur.getText());
        pst1.setString(2, Interface_sa.parking.getText());
        pst1.setString(3, Interface_sa.numeroplace.getText());
        pst1.setString(4, Interface_sa.prix.getText());
        pst1.setString(5, Interface_sa.numerocarte.getText());
        SimpleDateFormat sdf= new SimpleDateFormat("yyy-MM");
        String date= sdf.format(Interface_sa.dateexpiration.getDate());
        pst1.setString(6, date);
```

Figure 5.5 : Exemple sur l'implémentation des comportements (paiement)

4. Description du système

Dans cette section on présente une description du système réalisé. Nous commençons par la présentation des différentes interfaces du système. Ensuite, nous allons donner une simulation de l'exécution d'un scénario avec les outils de Jade.

4.1. Les interfaces principales du système

4.1.1. Interface d'inscription

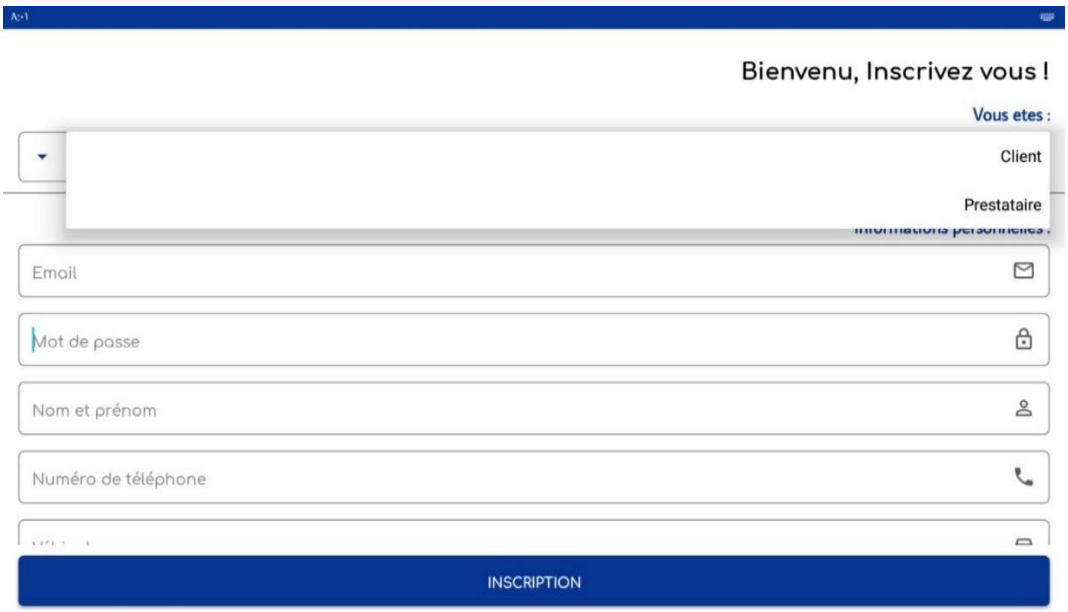


Figure 5.6: Interface inscription

4.1.2. Interface authentification(login)

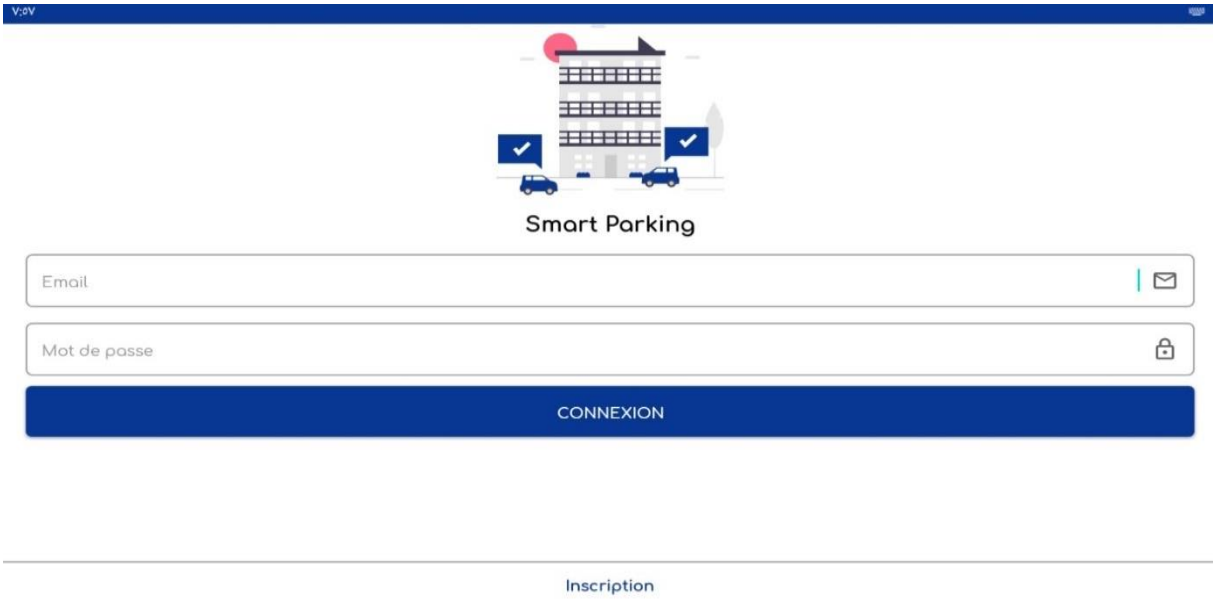


Figure 5.7: Interface authentification

4.1.3. Interfaces pour administrateur du système

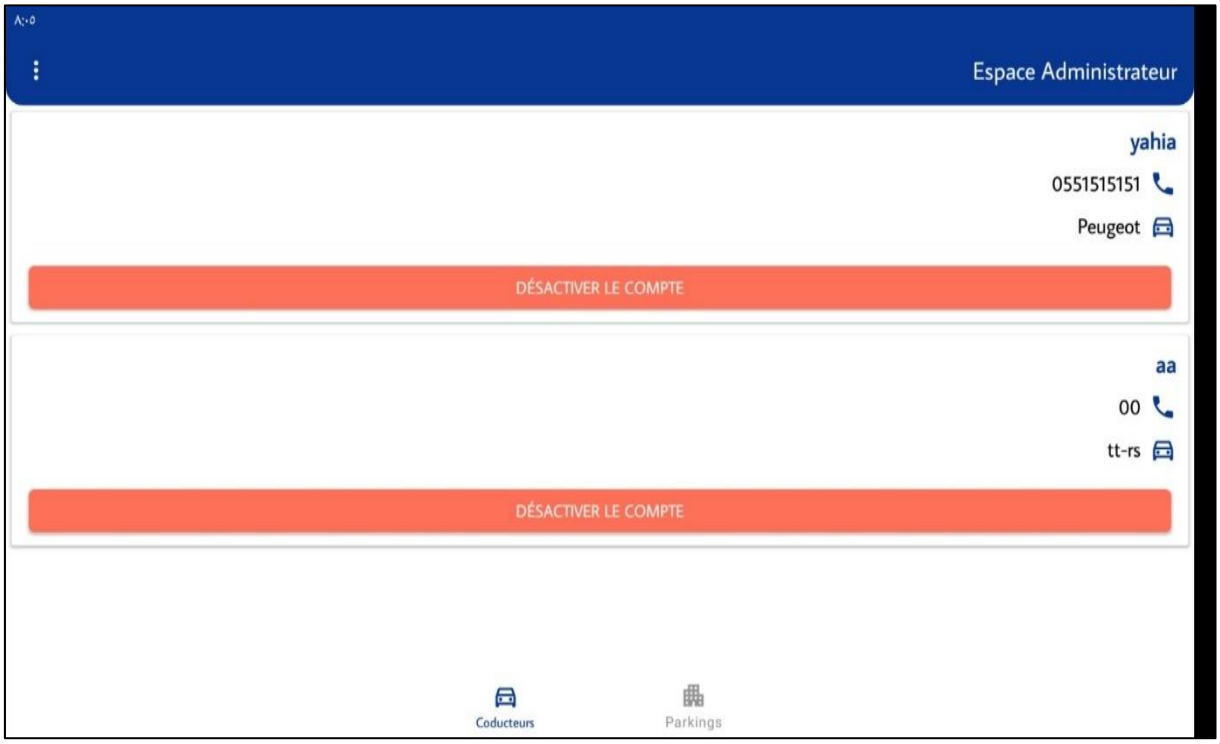


Figure 5.8 : Gérer les comptes (compte conducteur)

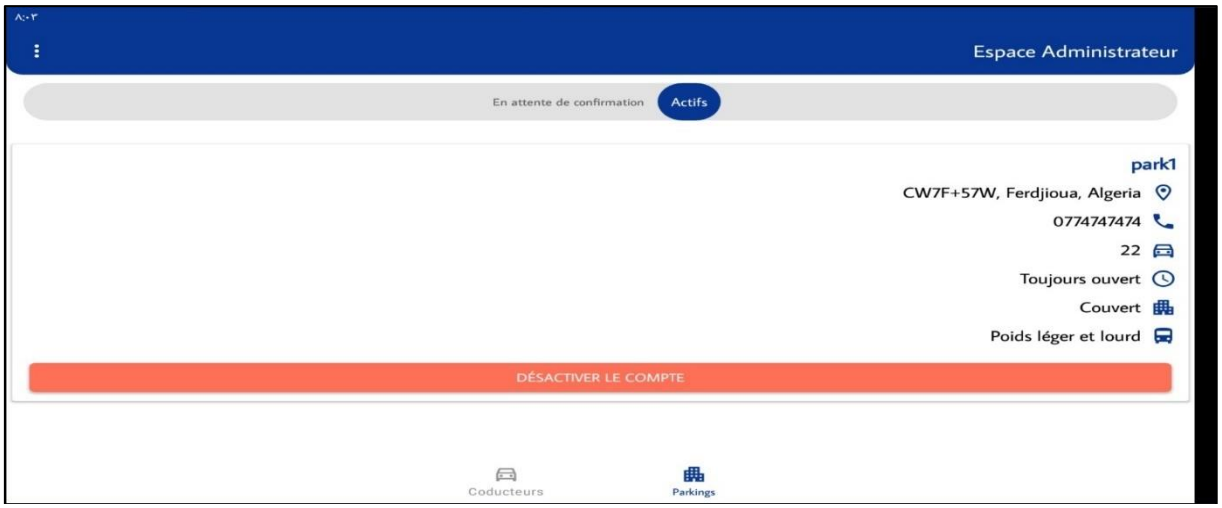


Figure 5.9: Gérer les comptes (compte prestataire « gestionnaire de parking »)

4.1.4. Interfaces pour les gestionnaires des parkings (parking manager)

Figure 5.10 : Etat de parking

Figure 5.11 : Mise ajour parking

Figure 5.12 : Gérer les réservations

4.1.5. Interfaces pour les conducteurs

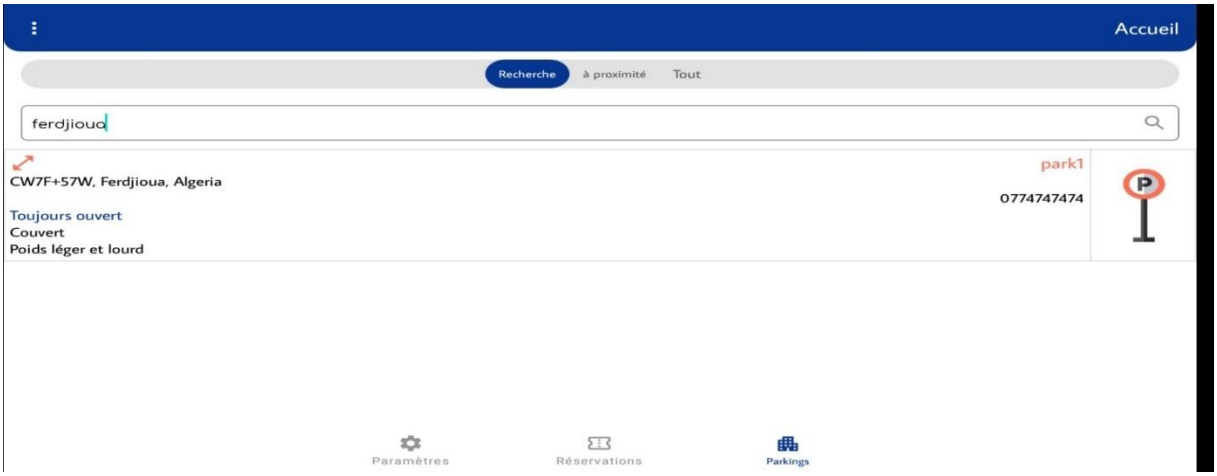


Figure 5.13 : Rechercher une place

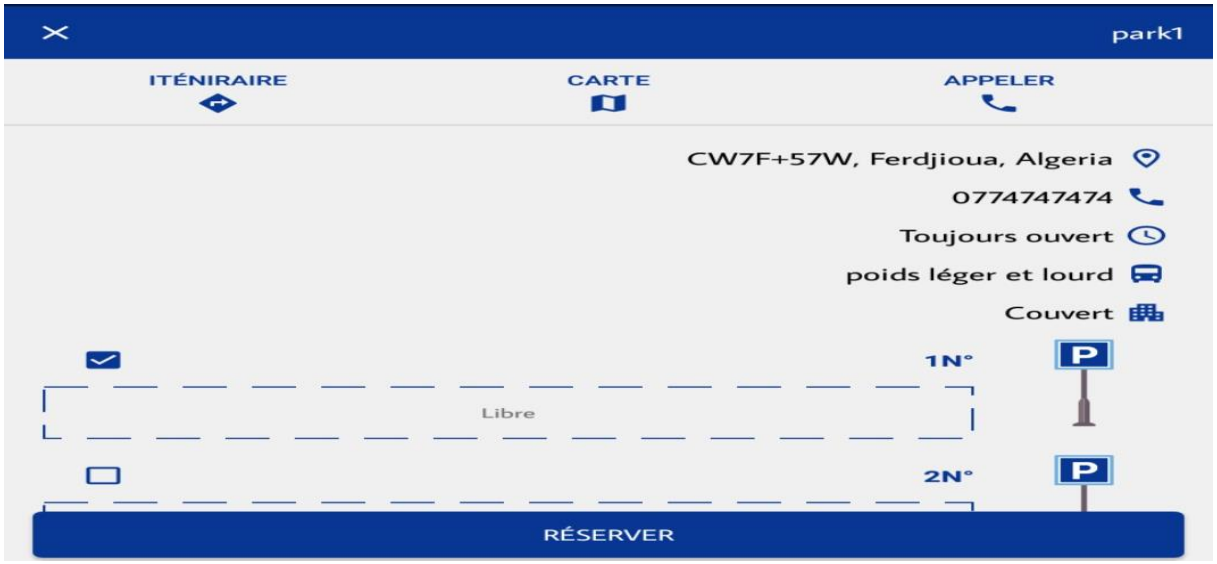


Figure 5.14 : Réserver une place

4.2. Les agents du système

La figure (5.15) correspond à l'interface graphique (GUI) de l'agent RMA (Remote MonitoringAgent) dont le rôle est de contrôler et superviser la plate-forme ; il est responsable de l'authentificationet l'enregistrement des agents du système. Cette figure montre les différents agentsde notre système ainsi que les différents conteneurs dans lesquelles ces agents s'exécutent.

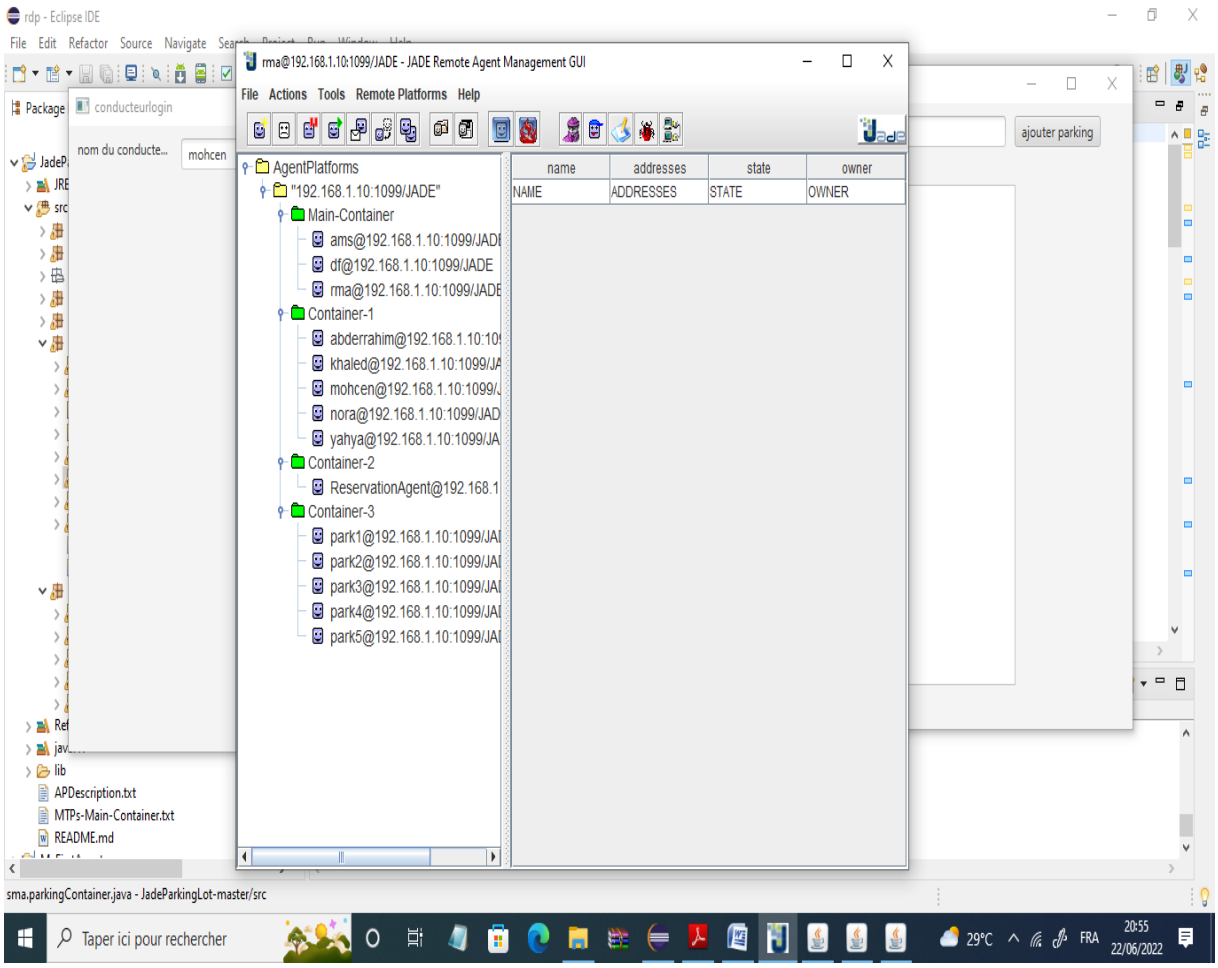


Figure 5.15 : Les agents du notre système présentés avec le GUI de l'agent RMA

Pour simuler l'interaction entre les différents agents de notre système nous avons utilisé. L'agent Sniffer du Jade. La figure (5.16) montre un exemple des interactions entre les agents du système.

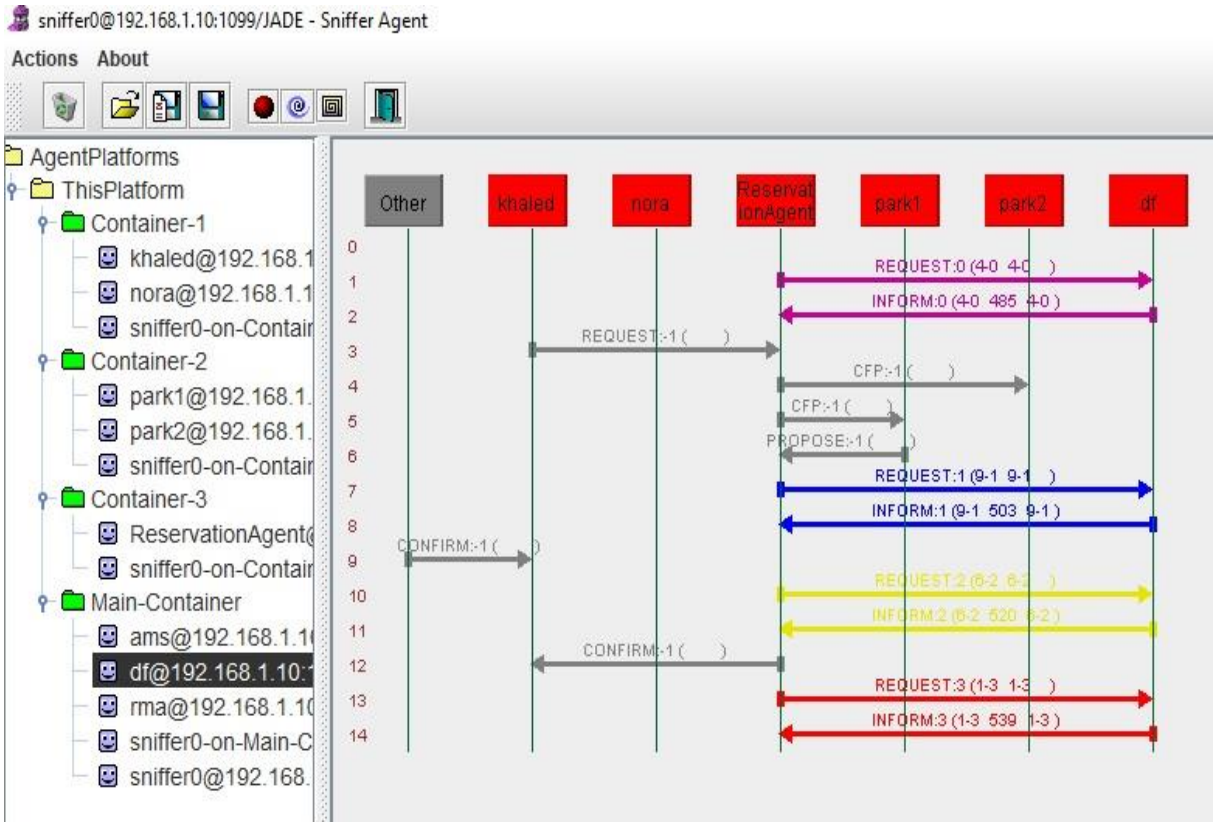


Figure 5.16 : Simulation des interactions entre agents par l'agent Sniffer (Rechercher une place + réservation)

5. Conclusion

Dans ce chapitre, nous avons présenté les différents outils, environnements ainsi que la plateforme de développement utilisés. Nous avons ensuite expliqué brièvement l'implémentation du système avec Jade. Dans la quatrième section nous avons présenté les différentes interfaces de notre système ainsi qu'une illustration des agents tels qu'ils sont dans la plateforme Jade et la communication entre eux.

Conclusion générale

Conclusion générale

La technologie continue à évoluer avec une vitesse incroyable. Notamment dans les domaines de l'information et de la communication, ainsi que les terribles évolutions dans le domaine des téléphones portables qui deviennent très fréquentes, sans parler de l'émergence de la technologie Internet des objets, ainsi que des systèmes multi-agents, qui ont entraîné des changements et des développements importants dans de nombreux domaines. Dans ce travail, nous mettons en évidence le domaine de la circulation et du stationnement, en particulier dans cette grave surcharge de trafic et la difficulté de trouver des places de stationnement en temps opportun.

L'objectif de notre étude était de proposer une solution efficace pour permettre, en particulier, aux conducteurs d'effectuer des opérations de recherche et de réservations des places de stationnement en utilisant leurs appareilles mobiles. Pour cela, nous avons réalisé dans ce travail un système intégré se basant sur la technologie mobile et sur les systèmes multi-agents, en plus de ça la gestion et la surveillance de parking basé sur l'IOT (smart parking).

Pour la réalisation de ce projet de fin d'étude nous avons suivi les principales étapes suivantes :

- En premier lieu, nous avons passé en revue les concepts de l'IOT. Nous avons extrait d'après cette étude les aspects à prendre en considération lors du développement de notre application.
- Ensuite, nous avons décrit les notions agent et systèmes multi-agents ainsi que leurs caractéristiques et leurs différents domaines d'application.
- Pour la conception de notre système, nous avons basé sur la méthode Voyelles et sur le langage AUML et ses différents diagrammes.
- En fin, pour la réalisation de notre système nous avons utilisé plusieurs plateformes, outils et langages de développement, telqu'Arduino, JADE, Java et Android.

Notre système de stationnement intelligent va aider à la résolution de problème de recherche des places de parking dans les grandes villes. Ce système va permettre en particulier :

- Aux gestionnaires de parkings : de publier leurs services, la gestion facile de leurs parkings et l'augmentation de la rentabilité.
- Aux conducteurs : de rechercher et de réserver des places de stationnement facilement.

Les résultats de ce travail peuvent constituer les bases d'un travail à poursuivre et à améliorer pour une étude beaucoup plus approfondie comme :

- L'utilisation d'une camera pour la detection et la reconaissance des matricules pour indiquer les véhicules ayant des reservations.
- L'utilisation des mécanismes de négociation entre les agents.
- L'intégration des agents avec des fonctionnalités spécifiques comme la sécurité et les agents assistants.

Glossaire

IOT: internet of thing.

CERP_Ido : Cluster des projets européens de recherche sur l'Internet des objets.

RFID : la Radio Frequency Identification est une méthode permettant de mémoriser et récupérer des données à distance.

LPWAN: Low-power Wide-area Network.

Les réseaux LPWAN sont une classe de technologie de réseau sans fil grande distance à basse consommation.

AWS : Amazon Web Services est une division du groupe américain de commerce électronique Amazon, spécialisée dans les services de cloud computing à la demande pour les entreprises et particuliers.

NetSA : (Networked Software Agents) Une architecture multiagent pour la recherche sur Interne de l'université laval canada.

CSELT : Groupe de recherche de Gruppo Telecom, Italie.

FIPA: Foundation for Intelligent Physical Agents.

Bibliographie

[1] « Les enjeux et les défis de l'Internet des Objets (IdO), » Internet des objets, vol. 1, n 11, 05 Avril 2017.

[2] « Internet des objets : qu'est-ce que c'est ? » [En ligne]. Disponible sur : https://www.futura-sciences.com/tech/definitions/internet-internet-objets-15158/. [Accès le 02 Juin 2022].

[3] EY, « Définitions autour des objets connectés, » 06 Mars 2017. [En ligne]. Disponible sur : http://www.smartgrids-cre.fr/. [Accès le 02 Juin 2022].

[4] Internet of Things (IoT) : Concepts, Issues, Challenges and Perspectives Laboratoire Paragraphe, Université Paris 8, imad.saleh@univ-paris8.fr. [En ligne]. Disponible sur : https://www.openscience.fr/IMG/pdf/iste_ido18v2n1_1.pdf [Accès le 15 Juin 2022].

[5] P. Bouffaron, « Les Objets Connectés : la nouvelle génération d'Internet ?» 13 Septembre 2013. [En ligne]. Disponible sur : https://www.france-science.org/. [Accès le 30 Mai 2022].

[6] Le microcontrôleur : Qu'est-ce qu'un microcontrôleur et à quoi sert-il ? Quel est son fonctionnement ? – 2022. Disponible sur :

https://clubifone.com/quest-ce-quun-microcontroleur-et-a-quoi-sert-il-quel-est-son-fonctionnement-2022/ [Accès le 30 mai2022].

[7] Qu'est-ce que Arduino ? [En ligne]. Disponible sur : https://www.positron-libre.com/electronique/arduino/arduino.php

[8] Capteur - Définition et Explications [En ligne]. Disponible sur :

https://www.techno-science.net/definition/3690.html.

[9] [Les actionneurs et les capteurs] [En ligne]. Disponible sur :

https://www.technologuepro.com/cours-automatismes-industriels/chapitre-2-les-actionneurs-et-les-capteurs.pdf.

[10] Capteur de température analogique LM35 avec Arduino [En ligne]. Disponible sur :https://www.volta.ma/capteur-de-temperature-analogique-lm35-avec-arduino/arduino.

[11] Capteur de mouvement PIR HC-SR501 et Arduino [En ligne]. Disponible sur :

https://www.robot-maker.com/shop/blog/27_Connecter-capteur-Infrarouge-HCSR501.html

[12] Actionneur : définition et fonctionnement de l'appareilLa RédactionJDN [En ligne]. Disponible sur :

https://www.journaldunet.fr/web-tech/dictionnaire-de-l-iot/1489525-actionneur-definition-et-fonctionnement-de-l-appareil/

[13] What is a Servomotor? [Enligne]. Disponible sur :

https://www.electrical4u.com/what-is-servo-motor/

[14] Protocoles IoT [Enligne]. Disponible sur :

https://iot.goffinet.org/iot_protocoles.html.

[15] L'environnement de développement (EDI) de l'ARDUINO - Audit & Systèmes d'Information (auditsi.eu) Disponible sur :

https://www.auditsi.eu/?p=7085.

[16] Bienvenue dans la documentation de Spyder Disponible sur :

https://docs.spyder-ide.org/current/index.html

[17] Qu'est-ce que AWS Cloud9 ? Disponible sur :

https://docs.aws.amazon.com/fr_fr/cloud9/latest/user-guide/welcome.html

[18] AWS IoT (Internet des objets Amazon Web Services) Disponible sur :

https://www.techtarget.com/searchaws/definition/AWS-IoT-Amazon-Web-Services-internet-of-things

[19] Google Cloud Platform Disponible sur :

https://fr.wikipedia.org/wiki/Google_Cloud_Platform

[20] Impact de l'IoT dans les transports Disponible sur :

https://iot.do/impact-iot-transportation-2016-10

[21] Smart Agriculture : Intelligente, collaborative et durable Available :

https://www.matooma.com/fr/s-informer/actualites-iot-m2m/smart-agriculture-intelligente-collaborative-et-durable

[22] E-santé : qu'est-ce que c'est ? Disponible sur :

https://www.futura-sciences.com/sante/definitions/medecine-e-sante-15728/

[23] 3 raisons d'opter pour la domotique dans sa maison Disponible sur :

https://www.lemagdeladomotique.com/dossier-2-exemples-domaines-application-domotique-maison.html

[24] Construction durable et villes intelligentes en France Disponible sur :

https://www.construction21.org/france/articles/h/construction-durable-villes-intelligentes-en-France.html

[25] 10 Benefits of a Smart Parking Solution Disponible sur:

https://www.plasmacomp.com/blogs/benefits-of-smart-parking-solution/

[26]: IOT architecture Available:

https://iotindustriel.com/tendances-de-liot-industriel/architecture-iot-lessentiel-a-savoir/

[27] SEFACENE Mohamed Lamine.,2016, MODÉLISATION ET RÉALISATION D'UN SYSTÈME D'ENCHERES PAR LES SYSTÈMES MULTI-AGENTS.MEMOIRE DE MASTERGénieLogiciel.Université Université Abderrahmane Mira de Béjaïa 1-7 p

[28] TOUMI Mehdi, DERABLIA Hichem.,2011, Automate cellulaire pour la simulation orientée agents. Mémoire de Master : Ingénierie des médias. Université de Guelma 1-67 p

[29] cours Agents et SMA Un support de cours destiné aux étudiants de 02ème année Master Option : STIC Université MILA Département des Mathématiques et Informatique

Préparé par : Dr. Nardjes BOUCHEMAL

[30] : J. KAMARA, "Conception et réalisation d'un système de gestion de véhicules partagées : de la multi-modalité vers la co-modalité", thèse, 2012.

[31] Tony Garnea.,2003, Un Environnement de Développement De Systèmes Multi-Agents Totalement Distribués.MEMOIRE de Licence : en mathématiques et informatique appliquées. À l'Université du Québec à Trois-Rivières 1-190 p

[32] MEMOIRE de fin d'études pour l'obtention du diplôme de
MASTER EN INFORMATIQUE Option : Architectures Distribuées
Présenté par : BENBOUDRIOU Manel chapitre2 page 23

[33] BENHAMZA KARIMA.,2016, Conception d'un système multi-agents adaptatif pour la résolution de problème, diplôme de Doctorat : Intelligence Artificielle, UNIVERSITE BADJI
MOKHTAR-ANNABA 1-166 p

[34] Cours Systèmes Multi-Agents. [Pdf], disponible sur : http://fmi.univ-tiaret.dz/images/1GL/2019.2020/s1/Chapitre-3---Cours-Systmes-Mu.pdf

[35] Ferber J. « Les Systèmes Multi-Agents Vers une intelligence collective ». Interéditions,1995.

[36] Mr. DJAABOUB Salim.,2009, Logique floue et SMA : Aide à la décision floue dans les systèmes multi-agents, Magister en informatique, UniversitaireMentouri de Constantine Faculté de science et de l'ingénierie Département Informatique 1-93 p

[37] YoussfiMohamed.SystemesMulti-Agents, Laboratoire : Signaux Systèmes Distribués et Intelligence Artificielle (SSDIA) ENSET Mohammedia, Université Hassan II Casablanca 1-113 p

[38] Un support de cours destiné aux étudiants de 02ème année Master Option : Architectures Distribuées. Préparé par Dr. Toufik MARIR Maître de Conférences – B– Département des Mathématiques et d'Informatique, Membre de l'équipe DISE, Laboratoire ReLa (CS)2. P27-50

[39] un support pdf Méthodes orientées agent et multi-agent : L'approche Voyelles page 19
Chapitre rédigé par Carole Bernon, Marie-Pierre Gleizes et Gauthier Picard.
Disponible sur : https://www.emse.fr/~picard/publications/bernon09industrie.pdf .. Consultée le 05/05/2022

[40] IR PROXIMITY SENSER disponible sur : https://www.makerlab-electronics.com/product/ir-proximity-sensor///consulté le 04/06/2022

[41] condensateur definition disponible sur :
https://fr.wikipedia.org/wiki/Condensateur

[42] Simulation de circuits et composants pilotés par Arduino : Tinkercad / Circuits

Disponible sur :

https://eduscol.education.fr/sti/ressources_techniques/simulation-de-circuits-et-composants-pilotes-par-arduino-tinkercad-circuits

[43] Qu'est-ce que la technologie Java et pourquoi en ai-je besoin [html], disponible sur : https : //www.java.com/fr/download/faq/whatis_java.xml

[44] La plate-forme JADE [html], disponible sur : http://turing.cs.pub.ro/auf2/html/chapters/chapter6/chapter_6_5_1.html.

[45] JADE PROGRAMMING FOR BEGINNERS Authors: Giovanni Caire (TILAB, formerly CSELT)

[46] What is JADE-LEAP and what are its relationships with JADE? disponible sur

https://jade.tilab.com/support/faq/what-is-jade-leap-and-what-are-its-relationships-with-jade/

[47] : Android Studio[html], disponible sur :
https://www.01net.com/telecharger/windows/Programmation/creation/fiches/129241.html

[48] Firebase Realtime Database disponible sur :
https://firebase.google.com/docs/database

[49] MySQL[html], disponible sur : https://sql.sh/sgbd/mysql

[50] JDBC – Introduction disponible sur :

https://web.maths.unsw.edu.au/~lafaye/CCM/jdbc/jdbcintro.htm

[51] : Desktop IDEs[html], disponible sur : https://www.eclipse.org/ide/