*Ministère de l'Enseignement Supérieur*

*Et de la Recherche scientifique*

**Université SAAD DAHLEB de BLIDA**

Faculté des sciences

**Département d'Informatique**

**Laboratoire : LRDSI**



**Mémoire de fin d'étude**

Présenté en vue d'obtenir le diplôme de Master

**Thème**

# Contrôle télérobotique MOSR basé sur la technologie WebRTC : Application au robot RobuTER/ULM

**Organisme d'accueil :** Centre de Développement des Technologies Avancées **(CDTA)**

C.D.T.A

Président: Mr. Farah

**Présenté par :**

- BOUROUBA Ilyes
- MASSAID Sami Aghiles

**Encadrants :**

- Mr. Tiberkak Allal
- Mr. Hentout Abdelfetah

**Promoteur:**

- Mr. Chikhi Fateh Nassim



**Année universitaire : 2017/2018**

# Remerciements

Nous tenons tout d'abord à remercier **Dieu** le tout puissant et miséricordieux, qui nous a donné la force et la patience d'accomplir ce modeste travail.

Nos remerciements s'étendent également aux **Dr. HENTOUT Abdelfetah** et **Dr. TIBERKAK Allal.** Pour nous avoir donné l'opportunité de travailler sur ce projet qui nous a passionnés. Et pour leur disponibilité et leur confiance. Et également pour leurs aides et leurs conseils. Qu'ils trouvent dans ce travail un hommage vivant à leurs hautes personnalités.

Nous souhaiterons manifester notre grande reconnaissance particulièrement à notre encadrant **Dr. CHIKHI Nacim Fateh**, pour l'orientation, la confiance, la patience qui ont constitué un apport considérable sans lequel ce travail n'aurait pas pu être mené au bon port. Qu'il trouve dans ce travail un hommage vivant à sa haute personnalité.

Que tous ceux qui nous ont aidés de près ou de loin, trouvent ici l'expression de notre gratitude.

# Dédicaces (SAMI)

Que ce travail témoigne de mes respects :

A mon père Hocine et à ma chère mère Karima qui m'ont toujours poussé et motivé dans mes études. Grâce à leurs tendres encouragements et leurs grands sacrifices, Je prie le bon Dieu de les bénir, de veiller sur eux, en espérant qu'ils seront toujours fière de moi

A mes frères amine, yacine, reda,ryad,imad.

A mes sœurs lina,nesrine.

A ma chère femme.

A ma tante malika et son marie kamel.

A toute la famille Lahbiben

Ils vont trouver ici l'expression de mes sentiments de respect et de reconnaissance pour le soutien qu'ils n'ont cessé de me porter

A tous mes Amis ILyes, Mohammed (ramy), islem, ihcen, faycel,Joe, amine, jimmy, yacine.

Ils vont trouver ici le témoignage d'une fidélité et d'une amitié infinie.

A mon binôme BouRouBa Ilyes.

A tous ceux qui m'ont aidé et soutenu pendant tout mon cursus universitaire.

# Dédicaces (ILYES)

Que ce travail témoigne de mes respects :

A mon père Nour-Eddine et à ma chère mère Noura qui m'ont toujours poussé et motivé dans mes études. Grâce à leurs tendres encouragements et leurs grands sacrifices, Je prie le bon Dieu de les bénir, de veiller sur eux, en espérant qu'ils seront toujours fière de moi

A mon frère Siff-Eddine.

A ma sœur Narimene.

A mon beau-frère Brahim

A mon neveu Khalil

Ils vont trouver ici l'expression de mes sentiments de respect et de reconnaissance pour le soutien qu'ils n'ont cessé de me porter

A tous mes Amis Yacine, Mohammed (ramy), aissam, Younes, ihcen, amine (minato), jimmy, joe.

Ils vont trouver ici le témoignage d'une fidélité et d'une amitié infinie

A Mon binôme Massaid sami aghiles

A tous ceux qui m'ont aidé et soutenu pendant tout mon cursus universitaire.

# ملخص

الهدف من هذا العمل هو السماح للمستخدمين بالتحكم والإشراف على روبوت عن بعد. يشير مفهوم التحكم عن بعد إلى أن تبادل المعلومات بين الروبوت والمتحكم يتم نقلها بوساطة وسيلة اتصال. ومن هذا المنظور فان الهدف من هذا العمل هو تطوير تطبيق ويب للتحكم المتعد عن بعد MOSR و في الوقت الفعلي للروبوت RobuterULM هذا التطبيق سيسمح للمستخدم تجنب التنقل و تنفيذ المهام من مواقع مختلفة، مع توفير ميزات متقدمة مثل التحكم المتزامن للروبوت من قبل عدة مستخدمين. و يستخدم تطبيق الويب المحقق، تكنولوجيات الويب في الوقت الفعلي التي تم دمجها مؤخرًا في متصفحات الويب. و لتحقيق ذلك، اخترنا حلاً يستند على لغة JavaScript. ان استخدام هذه اللغة يجعل من الممكن الحصول على كود فعال إما على جانب العميل أو على جانب الخادم مما يضمن أداء عالي. كما استخدمنا ايضا Node.js لتطوير الخادم. و WebSocket لاتصال ثنائي الاتجاه بين العميل و الخادم. و أخيرا تكنولوجيا WebRTC للتواصل واحد لواحد بين مختلف كيانات النظام.

# Résumé

L'objectif de ce travail de Master est de permettre à des utilisateurs de contrôler et de superviser un robot à distance. La notion de contrôle à distance (ou télérobotique) implique que l'échange d'informations entre le robot et l'opérateur humain soit véhiculé par un moyen de communication. C'est dans cette optique que s'inscrit ce travail ; il s'agit de développer une application web pour le contrôle MOSR à distance et en temps réel du robot manipulateur mobile RobuTER/ULM. Une telle application permettrait d'éviter à l'utilisateur de se déplacer et de manœuvrer des tâches de différents endroits, tout en offrant des fonctionnalités avancées telles que le contrôle simultané du robot par plusieurs utilisateurs. L'application web réalisée utilise les technologies web temps réel incorporées récemment dans les navigateurs web. Pour la réalisation, nous avons opté pour une solution basée sur *JavaScript*. L'utilisation de ce langage permet d'avoir un code léger que ce soit du côté client ou du côté serveur assurant ainsi de hautes performances. Nous avons aussi utilisé *Node.js* pour la mise en œuvre des serveurs, les *WebSockets* pour la communication client/serveur bidirectionnelle et enfin les *API* de *WebRTC* pour la communication en *peer-2-peer* entre les différentes entités du système.



**Mots clés :** WebSocket, socket.io, WebRTC, JavaScript, Node.js, application distribuée, télérobotique, téléopération, MOSR, communication en temps réel.

# Abstract

The goal of this project is to allow users to control and supervise a remote robot. The concept of remote control (or telerobotic) implies that the exchange of information between the robot and the human operator is conveyed by a means of communication. It is in this perspective that this work fits. The aim is to develop a web application for remote and real-time MOSR control of the RobuTER / ULM mobile manipulator robot. Such an application would prevent the user from moving and maneuvering tasks from different locations, while offering advanced features such as simultaneous control of the robot by multiple users. The realized web application uses real-time web technologies recently incorporated into web browsers. For the realization, we opted for a solution based on JavaScript. The use of this language makes it possible to have a light code either on the client side or on the server side thus ensuring high performances. We also used Node.js for server implementation, WebSockets for bidirectional client / server communication, and finally WebRTC APIs for peer-to-peer communication between the different entities of the system.

**Keywords:** WebSocket, socket.io, WebRTC, JavaScript, Node.js, distributed application, telerobotic, remote operation, MOSR, real-time communication.

# Table des matières

Chapitre 3 : Conception du système de contrôle télérobotique

# Liste des figures

# Liste des tableaux

# Introduction générale

Les robots ont été incorporés dans la vie quotidienne au cours du dernier demi-siècle, ce qui n'était autrefois que de la science-fiction est maintenant devenu une réalité, Aujourd'hui, tout le monde bénéficie des progrès de la robotique dans la vie quotidienne.

Avec le développement des technologies le contrôle à distance des robots est devenu une nécessité, et donc un nouveau terme est apparu « télé-robotique », ce concept d'opération robotique à distance à apporter de nombreux avantages dans plusieurs domaines différents. Désamorcer les bombes, arpenter l'espace et la mer profonde, et traiter les patients sur le champ de bataille depuis un havre de sécurité derrière le front ne comprennent que quelques applications potentielles, ou encore sauver des vies humaines lord des catastrophes naturelles. Mais le problème ici c'est que la plupart des solutions existante nécessite des matériels bien précis pour les manipulations à distance, et l'absence du travail collaboratif sur le robot, ou encore ne traite pas la communication en temps réel entre le robot et l'esclave (celui qui contrôle le robot à distance), ce qui peut poser plusieurs problèmes.

Par conséquent dans notre projet nous allons y remédier à cela et d'apporter une contribution à ce vaste domaine en développant une application web pour le contrôle d'un robot en MOSR (plusieurs opérants qui contrôle un seul robot) en se basant sur la technologie WebRTC (web en temps réel), cette dernière offre de nombreux avantages tels que assurer la communication en temps réel entre les différents entités su notre systèmes et la réduction de la charge sur le serveur d'applications, l'augmentation de la quantité de données qui peuvent être traitées par le serveur, etc.

Pour assurer une meilleure présentation du notre travail effectué et garantir la clarté du notre mémoire, outre cette introduction générale, ce manuscrit se compose de quatre chapitres, une conclusion générale, chacun met en évidence une contribution particulière du travail :

- **Chapitre I :** Dans ce chapitre, nous présente l'état de l'art des systèmes de téléopération existants avec leurs différentes composantes. et décrit leurs principaux objectifs et présente leurs fonctionnalités ainsi que leurs avantages et inconvénients.
- **Chapitre II :** Nous présentons dans ce chapitre la technologie web en temps réel(WebRTC) que nous allons utiliser pour la mise en œuvre de notre application. Nous donnons sa définition et nous expliquons son fonctionnement, ainsi que ces différentes APIs.
- **Chapitre III :** Dans ce chapitre, nous décrivons l'étude analytique et conceptuelle de notre application en utilisant le langage UML.
- **Chapitre IV :** Dans ce chapitre, nous décrivons la partie la plus importante de notre travail qui concerne l'implémentation de notre application web en utilisant plusieurs technologies web, avec quelques tests effectués et les résultats obtenus pour illustrer les différentes fonctionnalités de notre application web.

# Chapitre 1

# État de l'art des systèmes télérobotique

# 1. Introduction

Les robots télé-opérés sont conçus pour permettre aux êtres humains de manœuvrer à distance les tâches dangereuses et sensibles par l'intermédiaire de robots assurant des meilleurs rapports de sécurité, des coûts faibles et des précisions meilleures. Comme représenté sur la figure 1, un système de téléopération présente généralement cinq composants [Keyvan, 95]:

- un opérateur qui exécute une tâche à distance.
- un contrôleur maître qui est interfacé à l'opérateur.
- un canal de communication par lequel des commandes de contrôle sont transmises.
- un robot esclave qui accomplit la tâche.
- un environnement à distance sur lequel la tâche est exécutée.

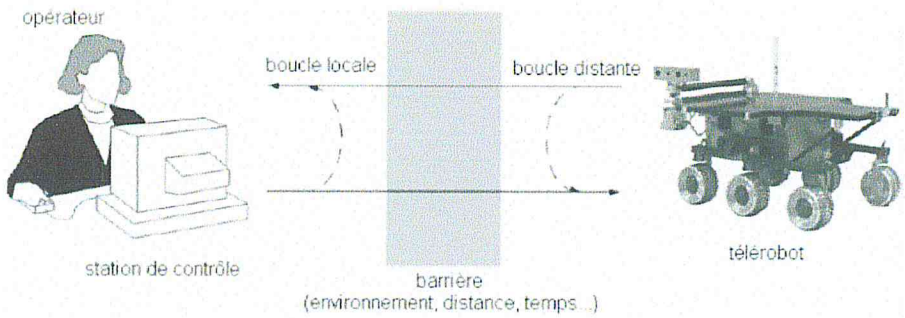


Figure 1 : Schéma d'un système de télérobotique [Keyvan, 95].

Basé sur l'interaction entre l'opérateur et le contrôleur maître, une commande (consigne) est transmise à l'esclave à travers le canal de communication afin d'exécuter la tâche désirée au niveau du robot distant. Afin d'empêcher des dommages, réduire le temps d'accomplissement de la tâche et augmenter les performances, l'information de contact de l'emplacement à distance est transmise à l'opérateur. Cette information peut être représentée à cet opérateur directement par un signal de retour au niveau du contrôleur maître ou bien, représentée indirectement à travers des afficheurs visuels ou acoustiques. L'information de contact sous forme de position mesurée ou de signal de force est convertie en un signal mécanique par l'actionneur au niveau du maître, puis, ce signal est présenté à l'opérateur humain.

## 2. Contrôle télérobotique via Internet

Ces systèmes utilisent Internet comme canal de communication par lequel les commandes de contrôle sont transmises aux robots. Dans ce qui suit, nous allons décrire quelques exemples de robots contrôlés via Internet.

### 2.1. Xavier

*Xavier* est un robot autonome capable de recevoir des commandes vocales (voir Figure 2). Sa caméra embarquée permet de visualiser ce que voit le robot [BCFLH, 98]. D'une part, l'opérateur peut choisir des tâches de haut niveau prédéfinies par le système (Simmons, 1998). D'autre part, le système retourne des informations sur la position et l'orientation toutes les 5 à 10 secondes ainsi qu'une image vidéo (de type GIF) toutes les 20 secondes. Néanmoins, le système ne supporte qu'un seul client (utilisateur) à chaque fois.

Figure 2 : Robot Xavier[BCFLH, 98].

### 2.2. Azkar

Le projet de recherche *Azkar* (« rapide » en basque) se concentre sur le contrôle à distance d'un robot mobile utilisant les technologies Web émergentes (WebRTC) pour la communication en temps réel. Le robot offre une immersion à distance grâce à ses capteurs (webcam, micros, lasers) qui permettent de visualiser et écouter les informations proposées dans le musée et d'interagir avec le robot guide [BZA, 16].

Grâce à l'utilisation du Web « pair-à-pair », la connexion entre le robot et son opérateur est rapide et robuste (grâce à un temps de latence réduit). Ceci permet d'obtenir une fluidité d'interaction améliorée. Par contre, comme pour le cas précédent, le système ne supporte qu'un seul utilisateur à la fois.

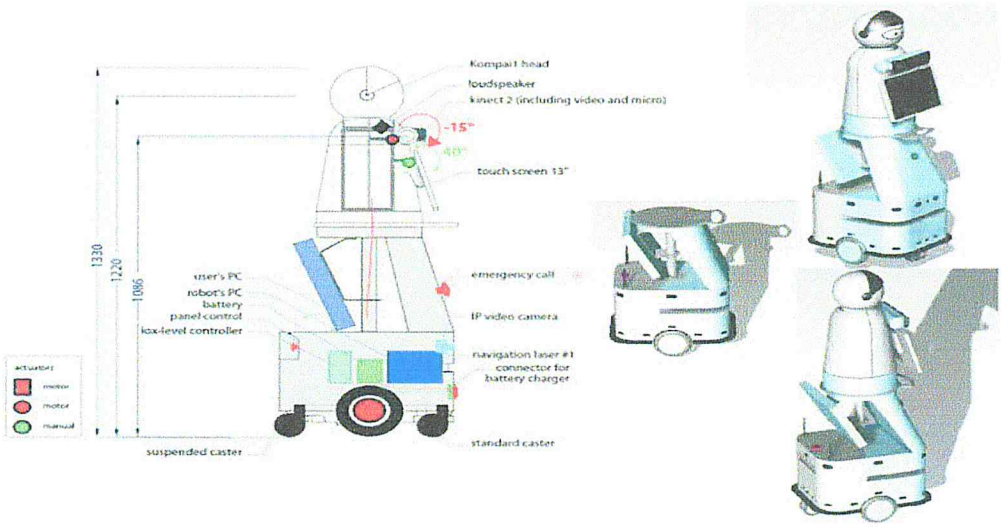


Figure 3 : Projet Azkar [BZA, 16].

## 2.3. Aleph

*Aleph* (Figure 4) est un manipulateur robotique de microscope télécommandé via Internet pour des sondes biologiques ou chimiques [ALP]. Son but est de permettre aux scientifiques de manipuler et d'analyser, en toute sécurité, des substances biologiques ou toxiques, ou simplement d'observer ou d'interagir avec de petites sondes et/ou dispositifs qui doivent être contenus dans des environnements contrôlés.

*Aleph* offre plusieurs avantages aux utilisateurs tels que la manipulation de sondes dans un environnement contrôlé très loin (le système est contrôlé via une connexion Internet) et avec des mouvements précis. Le robot permet aussi à l'utilisateur d'avoir un retour vidéo des systèmes (deux ou trois webcams). En revanche, *Aleph* ne supporte pas la communication en temps réel et il présente une marge de quelques secondes de retard dans la capture de la vidéo et dans l'exécution des commandes.

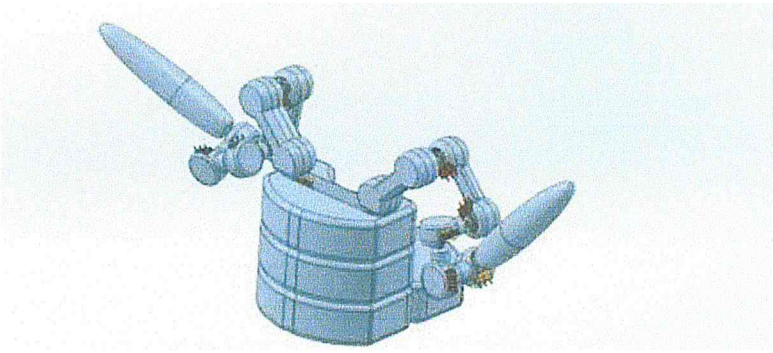

Figure 4 : Projet Aleph [ALP].

## 2.4. NeuroArm

*NeuroArm*, illustré par la figure 5, est un dispositif assisté par ordinateur, guidé par l'image et compatible avec l'IRM. Ce robot est conçu spécifiquement pour la neurochirurgie [PMSGLS, 09]. Il s'agit du premier robot de ce type destiné à la microchirurgie et à la stéréotaxie. Ce robot améliore la précision et la sécurité de toute intervention chirurgicale au cerveau, il permet aussi au chirurgien de voir en temps réel l'état de la liaison, le cerveau ainsi que l'emplacement des instruments chirurgicaux les uns par rapport aux autres.

L'un des avantages de *NeuroArm* est sa capacité à transmettre une image du cerveau en temps réel durant l'opération. Aussi, il possède une conception qui permet un échange rapide d'outils minimisant ainsi la perturbation du rythme chirurgical et les risques de dommages de ces outils. Cependant, l'un des problèmes majeurs de *neuroArm* est qu'il ne peut être manipulé que par un seul utilisateur à la fois.

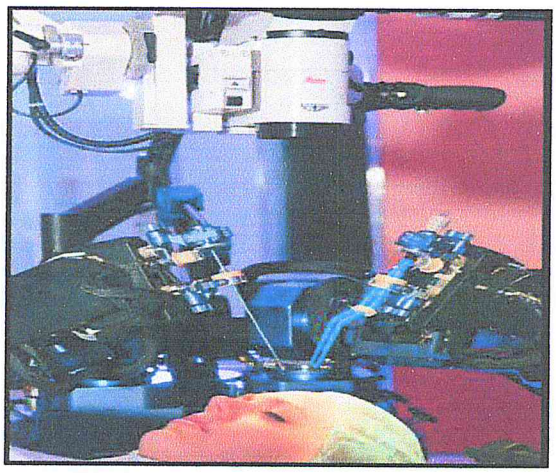

Figure 5 : Robot NeuroArm [PMSGLS, 09].

## 2.5. iRobot PackBot 510

PackBot 510 (figure 6) est un robot mobile tactique à multi-missions [Irobot,03]. Il est conçu pour être exploité par des militaires pour effectuer des missions dangereuses dans des champs de bataille à haut risque : surveillance et reconnaissance, détection chimique, biologique, radiologique et nucléaire (CBRN), dégagement de bâtiments et routes, neutralisation des explosifs, manipulation des matières dangereuses, détection des engins explosifs improvisés, etc. De plus, ce robot peut être configuré en fonction des besoins de la mission.

Le PC embarqué de RobuTER fonctionne avec le système d'exploitation Linux RedHat 9.0, qui a l'avantage d'être libre, ouvert, sous licence GPL - General Public License, et très bien documenté avec tous les outils nécessaires au développement du RobuTER lui-même.

Le développement d'applications pour RobuTER est basé sur Robosoft Development Toolchain. Ce développement est basé sur l'environnement de CAO SynDEx.

### 2.7. Discussion

Lors de l'étude des travaux cités précédemment, nous avons remarqué qu'en dépit que ces robots accomplissent bien leurs tâches, quelques failles peuvent être repérées. Pour le robot *Xavier*, le taux de communication entre l'utilisateur et le robot est considéré comme élevé, c'est le même que nous pouvons dire pour le cas du robot *Aleph,* ou encore *Azkar* qui ont tous une composante qui ne supporte pas une bonne qualité de vidéo, et on constate aussi que tous ces robots nécessitent des installations logicielles ou des outils spécifiques (des manettes) pour les commander ou bien les utiliser. On remarque aussi qu'ils sont tous manipulés par un seul opérateur ce qui peut induire des pertes de temps lors de l'exécution des tâches, ce qui est contreproductif.

Au terme de cette analyse, nous allons proposer un mécanisme qui va éliminer la plupart de ces failles. Nous allons, par conséquent, développer un système qui se basera sur les technologies web pour faciliter l'accès au robot et qui va assurer une communication en temps réel entre les différentes entités du système, tout cela grâce à l'exploitation de la technologie WebRTC.

## 3. Différents modes de contrôle télérobotique

Les modes de contrôle des systèmes télérobotiques communs peuvent être classifiés en quatre catégories [CKOKMT, 2000]:

### 3.1. Single Operator Single Robot (SOSR)

Un utilisateur humain téléopère ou contrôle un seul robot à la fois. Cela signifie une interaction entre un seul utilisateur et un seul robot *1-à-1*. De plus, c'est le type le plus répandu parmi les systèmes de téléopération (voir figure 8) [CKOKMT, 2000].

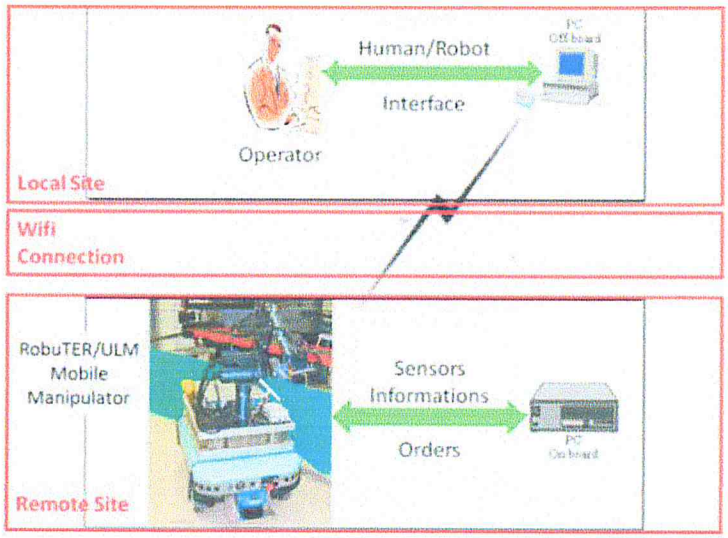


Figure 8 : Système de téléopération SOSR [CKOKMT, 2000].

### 3.2. Single Operator Multiple Robot (SOMR)

Un seul utilisateur contrôle plusieurs robots à la fois (Figure 9). Ce type de système appartient aux interactions de genre un-à-plusieurs (*1-à-n*) [CKOKMT, 2000].

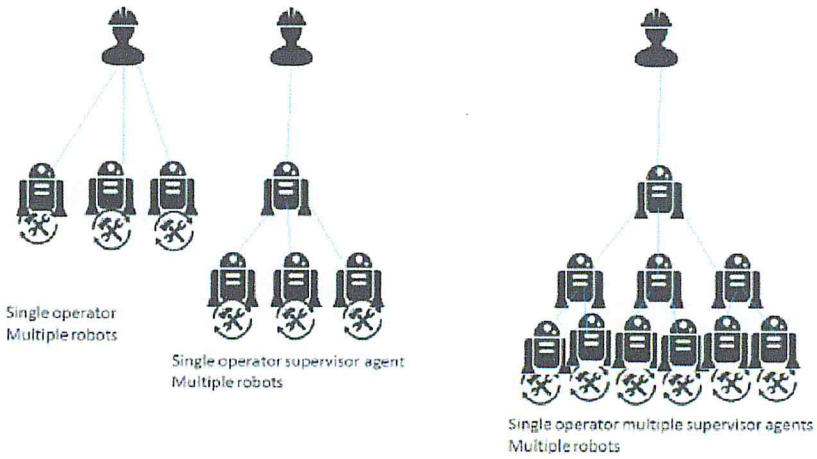


Figure 9 : Système de téléopération SOMR [CKOKMT, 2000].

### 3.3. Multiple Operator Multiple Robot (MOMR)

Chaque opérateur contrôle plusieurs robots et les robots ont des espaces de travail qui se chevauchent. Ceci est une interaction *n-à-m* mais quand $n=m$, cela signifie qu'il existe $n$ interactions de genre *1-à-1* (Figure 10) [CKOKMT, 2000].

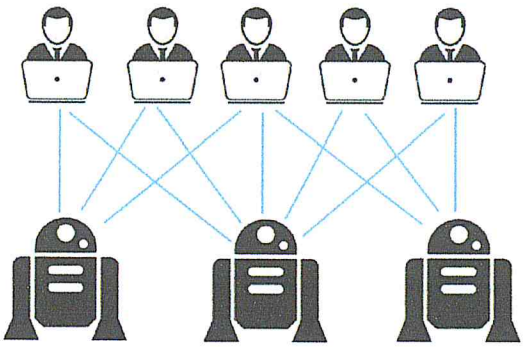

Figure 10 : Système de téléopération MOMR [CKOKMT, 2000].

### 3.4. Multiple operator Single Robot (MOSR)

Dans ce mode, plusieurs opérateurs contrôlent un seul robot (Figure 11). Cette catégorie se classe parmi les interactions de genre plusieurs-à-un (*n-à-1*) [CKOKMT, 2000].

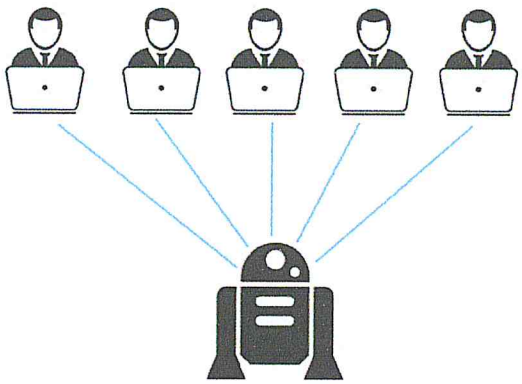

Figure 11 : Système de téléopération MOSR [CKOKMT, 2000].

## 4. État de l'art sur les systèmes MOSR

L'une des raisons qui ont poussé les travaux de *MOSR* sont les différentes études de l'interaction homme-homme dans les tâches de manipulation d'objets conjoints. Il a été montré que la performance de plusieurs humains à résoudre une tâche en collaboration est supérieure à celle d'un seul opérateur effectuant la même tâche [RP, 08]. Parmi les travaux réalisés dans cette catégorie (MOSR), les plus importants sont :

### 4.1. Da Vinci

Le système *Da Vinci* est composé de trois éléments principaux (figure 12). Tout d'abord, des moniteurs vidéo fournissent des images du site chirurgical ainsi que des informations utiles pour les médecins assistants. Puis, une table d'opérations équipée d'une console où sont fixés trois bras robotiques munis d'instruments chirurgicaux. La particularité

des instruments utilisés par *Da Vinci* est qu'ils disposent d'une extrémité qui possède deux degrés de liberté (ddl) supplémentaires, permettant d'augmenter significativement leur maniabilité. Enfin, le chirurgien est assis à une autre console depuis laquelle il peut contrôler les instruments chirurgicaux [GLM, 10].

Figure 12 : le robot Da Vinci [GLM, 10].

Pour ce qui est des avantages de ce robot, le système Da Vinci est caractérisé par les performances suivantes :

- Ses bras manipulateurs possèdent 7 ddl.
- La communication se fait en un temps quasi-réel.
- Le robot offre un contrôle de collaboration entre plusieurs opérateurs.

Bien que Da Vinci offre plusieurs avantages par rapport aux robots traditionnelles, il présente quelques inconvénients qui limitent son utilisation :

- Les interventions réalisées avec le Da Vinci ont un coût très élevé.
- La plateforme du robot est difficile à comprendre.
- La qualité vidéo est moyenne.
- L'accessibilité au robot se fait via un dispositif spécifique (console de commandement).
- La distance entre le robot et le chirurgien ne doit pas dépasser quelques mètres.

### 4.2. Zeus

Les bras robotiques du système *Zeus* sont directement fixés à la table chirurgicale (figure 13). Ceux-ci peuvent être mobilisés selon trois ddl, ce qui signifie que *Zeus* présente deux ddl en moins que *Da Vinci,* donc il est plus difficilement maniable. En ce qui concerne la position du chirurgien, celle-ci est également localisée au niveau d'une console depuis laquelle il contrôle à distance les bras manipulateurs. En outre, le chirurgien bénéficie d'une

image 3D, mais elle ne peut être vue sur un écran que via le port d'une paire de lunette spéciale. Ce système a néanmoins fait ses preuves lors de l'opération *Lindbergh* en septembre 2001 qui constitue, à ce jour, l'expérience télé-chirurgicale effectuée sur la plus longue distance. Le système *Zeus* offre comme avantage [SG, 01]:

- Les robots chirurgiens sont équipés d'une vision 3D et d'un zoom, ceci représente un avantage évident car cela permet d'avoir une meilleure vision de l'opération avec une image plus nette.
- Il permet la réduction de la fatigue pour les opérations à longues durées.
- La Rapidité de l'exécution des taches.
- Stabilité de l'image rendue.

Néanmoins, le projet *Zeus* présente quelques inconvénients, entre autres :

- La plateforme du robot est difficile à comprendre.
- Nécessite des outils spécifiques.
- Un système de feedback moyen.
- Le temps d'installation des outils est considérable.

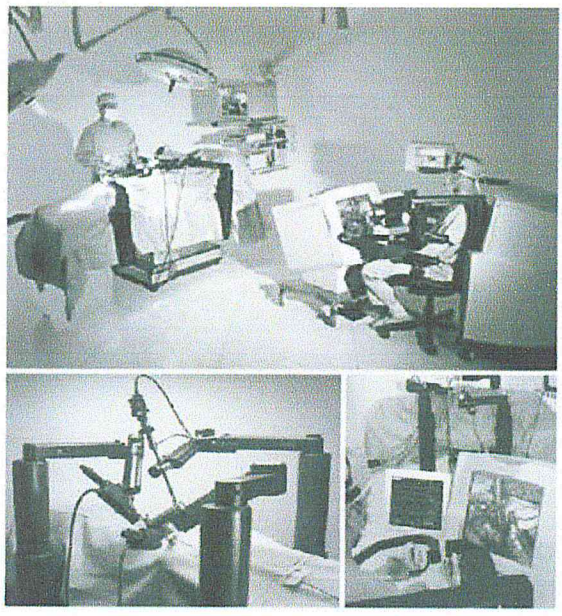

Figure 13 : Le système Zeus lors de l'opération chirurgicale transatlantique [SG, 01].

### 4.3. OCTOPUS

Le robot OCTOPUS a 26 ddl (figure 14) ; il a été développé pour réaliser des travaux complexes d'intervention en cas de catastrophe. OCTOPUS a un mode de contrôle MOSR [KTC, 17].

Parmi ces nombreux avantages on peut citer :

- Le contrôle se fait en collaboration entre deux opérants.
- La vitesse d'exécution des tâches est quasi-réelle.
- Possède des systèmes de caméra CCD (dispositif à transfert de charge).

Malgré ces avantages, le robot OCTOPUS présente plusieurs inconvénients, entre autres :

- L'avant et l'arrière des flippers sont difficilement contrôlables avec précision en même temps par un seul opérateur.
- Le contrôle du robot est manuel et pas à distance.
- Ne supporte pas plus de deux utilisateurs en même temps.
- L'opérateur n'a pas la possibilité de contrôler plusieurs articulations en même temps.

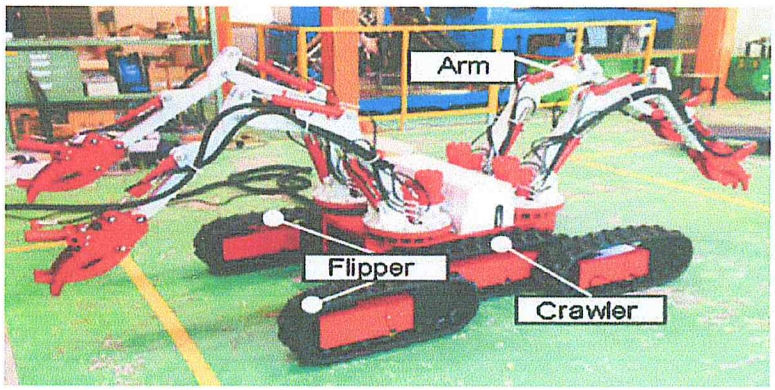


Figure 14 : Robot OCTOPUS [KTC, 17].

### 4.4. Discussion

En analysant les travaux précédents, nous constatons que ces projets bien qu'ils soient parvenus à répondre aux besoins fixés, cela n'empêche pas de relever quelques défaillances. Par exemple, pour le cas du robot *OCTOPUS* où deux opérateurs seulement peuvent le contrôler, ou encore l'impossibilité de l'opérateur à contrôler plusieurs articulations en même temps. Ou encore le robot Da Vinci qu'on ne peut contrôler qu'à une distance limitée.

En prenant en considération ces inconvénients, nous allons proposer une architecture générique de l'interaction MOSR. Ceci va permettre un contrôle multi-opérateurs du robot distant. Par conséquent, les opérateurs peuvent contrôler divers aspects du robot. De plus, ces opérateurs seront classés selon leur niveau d'accès (priorité).

D'autant plus d'après notre étude des systèmes télérobotiques existants on constate que le contrôle MOSR nécessite une collaboration entre les utilisateurs pour l'exécution des taches. C'est ce qui a motivé l'utilisation de la technologie WebRTC, parce qu'elle permet un bénéfice énorme pour tout ce qui est solutions de travail collaboratif. Cette technologie permet en effet de mettre en relation plusieurs utilisateurs en même temps sans ralentir la connexion, et ses possibilités d'intégration sont illimitées. Cela s'avère très pratique, car contrairement aux autres solutions, avec WebRTC tout se passe dans le navigateur et il n'y a rien à installer pour pouvoir utiliser le système.

## 5. Conclusion

Dans ce chapitre, nous avons donné un aperçu sur les systèmes télérobotiques existants avec leurs différentes composantes, ainsi que les applications multiples de ces systèmes. Nous avons vu comment l'utilisation d'Internet a réussi à rendre accessibles de nombreux systèmes robotiques, permettant ainsi la collaboration entre plusieurs opérateurs.

On retient de cette étude le manque des systèmes télérobotiques qui prennent en compte tous les aspects concernant le contrôle MOSR et la communication en temps réel et aussi l'accessibilité aux outils du contrôle de robot.

Le chapitre suivant est dédié à la description de la technologie web temps réel (WebRTC) utilisée dans le cadre de notre projet.

# Chapitre 2
# Technologies Web Temps Réel (WebRTC)

## 1. Introduction

*WebRTC* ou « Web Real-Time Communication » signifie littéralement « *Communication en temps réel pour le Web* ». C'est une interface de programmation (API) JavaScript développée au sein du W3C (World Wide Web Consortium) et de l'IETF (Internet Engineering Task Force). On parle également de canevas logiciel, ayant des implémentations au sein de différents navigateurs web, qui permet une communication en temps réel.[SAF, 18].

L'objectif de WebRTC est de lier des applications (VoIP, partage de fichiers *peer-to-peer*) tout en s'affranchissant des modules d'extensions propriétaires.

## 2. Origines de WebRTC

L'idée de WebRTC est née fin 2009, un an après le lancement de Google Chrome [SAF, 18]. En répertoriant les fonctionnalités manquantes dans son offre Web, Google a constaté que la plupart d'entre elles font déjà l'objet de projets, mais qu'il n'y a pas de solutions pour les communications en temps réel (RTC).

En 2011, à l'initiative de Google qui ouvre les sources des technologies de GIPS (Global Investment Performance Standards), des groupes de travail au W3C pour la standardisation d'une solution de RTC sur le Web ont alors été créés rapidement. D'autres grands acteurs comme Mozilla, Voxeo, Ericsson, Intel, Cisco, Samsung, France Telecom, AT&T, Avaya, Huawei et Opera participent à la naissance du standard WebRTC. Depuis, et par itérations successives, l'écriture du standard avance [WRTC, 17].

## 3. Architecture de WebRTC

WebRTC étend la sémantique client-serveur en introduisant un paradigme de communication peer-to-peer entre les navigateurs. Le modèle d'architecture WebRTC le plus général, donné par la Figure 15, s'inspire de ce que l'on appelle le trapèze SIP (Session Initiation Protocol).

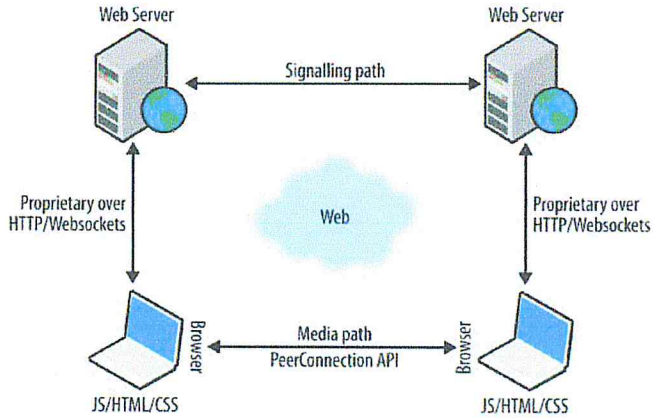


Figure 15 : Architecture du WebRTC (modèle du trapèze) [SAF, 18].

Dans le modèle donné par la figure 16, les deux navigateurs exécutent une application Web qui est téléchargée à partir d'un autre serveur Web. Les messages de signalisation sont utilisés pour établir et terminer les communications. Ils sont transportés par le protocole HTTP ou WebSocket via des serveurs Web qui peuvent les modifier, les traduire ou les gérer selon les besoins. Il est à noter que la signalisation entre le navigateur et le serveur n'est pas standardisée dans WebRTC, car elle est considérée comme faisant partie de l'application (voir Signalisation). En ce qui concerne le chemin de données, un *PeerConnection* (voir webrtc API) permet aux medias de circuler directement entre les navigateurs sans intervention de serveurs. Les deux serveurs Web peuvent communiquer en utilisant un protocole de signalisation standard, sinon, ils peuvent utiliser un protocole de signalisation propriétaire. Le plus courant scénario WebRTC est probablement celui où les deux navigateurs exécutent la même application, dans ce cas, le trapèze devient un triangle [SAF, 18].

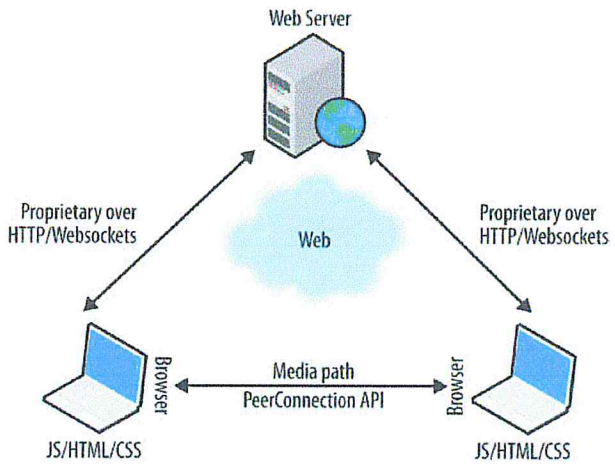


Figure 16 : Architecture du WebRTC (modèle du triangle) [SAF, 18].

## 4. API du WebRTC

WebRTC est un projet open source et gratuit qui permet aux différents navigateurs web et Smartphone de communiquer en temps réel via des APIs simples. Pour le développeur d'applications WebRTC, le standard comprend trois APIs JavaScript [SAF, 18]:

- *MediaStream* (API de flux réseau) : elle est également appelée *getUserMedia()*. Elle gère le flux de données audio/vidéo en provenance de la caméra ou du micro de l'utilisateur. Cette API prévoit le streaming de données capteurs, de fichiers, l'enregistrement des flux sur disque, etc.
- *RTCPeerConnection* (API de connexion) : elle permet à plusieurs utilisateurs de communiquer via leurs navigateurs. C'est l'API qui permet une communication stable et efficace de flux audio/vidéo entre pairs. Pour le développeur JavaScript, *RTCPeerConnection* cache tous les détails liés au streaming de la vidéo (réduction de bruit et d'écho, adaptation à la bande passante, codecs, etc.).
- *RTCDataChannel* : elle est similaire à WebSocket de JavaScript. Elle est utilisée pour envoyer des données non-medias à travers le réseau pour échanger des données. Au sein d'un canal de données, les applications peuvent transmettre des messages de façon ordonnée ou désordonnée. Un flux de données est créé lorsque l'un des pairs appelle la méthode *CreateDataChannel()* pour la première fois après avoir créé un objet *PeerConnection*. Chaque appel suivant à *CreateDataChannel()* créera un nouveau flux de données au sein de la connexion SCTP (Stream Control Transmission Protocol) ou protocole de transport équivalent dans un certain sens au TCP ou à l'UDP (Couche transport) existante [ACTS, 14].

## 5. Principes de fonctionnement

### 5.1. Composants de WebRTC

WebRTC se compose de deux parties, une première partie *Cliente* et une deuxième partie *Serveur* :

- *Partie « Cliente »* : elle est le plus souvent une application web tournant dans un navigateur, codée en JavaScript/HTML/CSS. Il existe également des SDK (Software Développement Kit) pour développer des clients natifs, en particulier, pour apporter des solutions WebRTC là où les navigateurs web ne supportent pas encore cette technologie (IOS, Safari, etc.).

- *Partie « Serveur »* : on trouve deux éléments distincts qui peuvent ne pas tourner sur la même machine :
  - *Serveur de signalisation* : il permet aux clients d'échanger leurs configurations (IPs publiques, type de medias, etc.).
  - *Serveur de « relais »* : il permet d'effectuer les échanges vidéo/audio/data lorsqu'une connexion P2P ne peut être réalisée directement entre deux clients. Les termes utilisés dans WebRTC sont « *serveur STUN* » pour la signalisation, et « *serveur TURN* » pour le serveur de relais.

Le cas le plus classique d'une application WebRTC comprenant une construction triangulaire et impliquant un serveur et deux clients est donné par Figure 17. Cette figure est un petit schéma qui montre l'établissement d'une connexion entre deux clients en utilisant WebRTC. A et B exécutent une application HTML5 à partir d'un navigateur web [JB, 12]:

1 : A demande au serveur une connexion avec B.

2 : le serveur relaie la demande de A à B.

3 : si B accepte, il envoie une demande de connexion à A.

4 : le serveur relaie la demande à A.

5 et 6 : les *PeerConnections* bidirectionnelles sont établies.

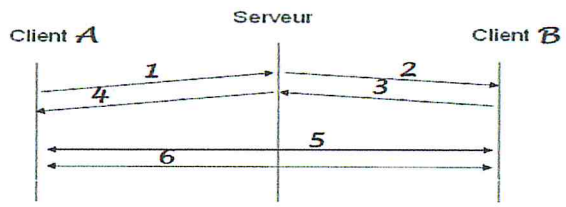


Figure 17 : Établissement d'une connexion entre deux clients en utilisant WebRTC.

## 5.2. Signalisation

La signalisation est le mécanisme par lequel les pairs s'échangent des messages de contrôle dans le but d'établir le protocole de communication, le canal et la méthode. Ceux-ci ne sont pas spécifiés dans le standard WebRTC. En fait, le développeur peut choisir n'importe quel protocole de message (comme *SIP* ou *XMPP*), et n'importe quel canal de communication duplex (comme *Websockets* ou *XMLHttpRequest*) en tandem avec une API de connexion persistante à un serveur *AppEngine*.

La signalisation sert à échanger trois types d'information : (*i*) les messages de contrôle de session (initialiser les communications et rapporter les erreurs), (*ii*) les configurations

réseau pour le monde « public », (adresse IP, ports, ...), et enfin (*iii*) les capacités « medias » (résolutions et codecs supportés par les browsers, etc.). On peut résumer le processus de signalisation en trois phases :

- Connexion au serveur de signalisation.
- Échange des informations réseau.
- Négociation des sessions media.

## 6. Framework ICE et protocoles STUN et TURN

Les applications WebRTC utilisent le framework ICE (Interactive Connectivity Establishment) pour surmonter les difficultés de la mise en réseau : la plupart des dispositifs en mesure d'utiliser WebRTC s'exécutent derrière une ou plusieurs couches de NAT (Network Address Translation) et peuvent avoir des couches de sécurité qui bloquent certains ports et protocoles parfois avec des DPP (Deep Packet Inspection ou inspection profonde de paquets IPs). De plus, beaucoup d'entre eux sont derrières des proxies et des firewalls d'entreprises, sans compter les firewalls et NAT des routeurs Wifi domestiques.

Le protocole STUN et son extension TURN sont utilisés par ICE pour traverser les NATs et se conformer aux aléas du réseau. ICE tente de connecter directement les pairs (les utilisateurs) sans se soucier des NATs (Figure 18). ICE tente d'abord d'établir une connexion directe entre les pairs, avec la plus faible latence possible, en utilisant l'adresse hôte obtenue à partir du système d'exploitation du pair et de sa carte réseau. [ST,18]

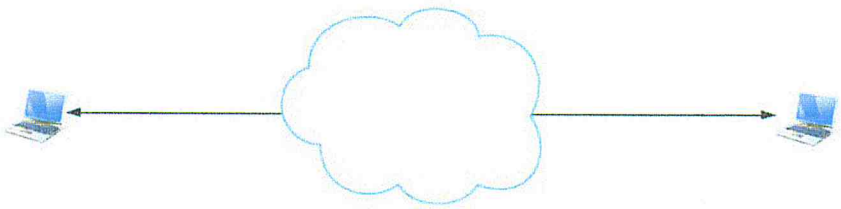

Figure 18 : Connexion directe entre pairs, sans NAT ni Firewall.

En cas d'échec (NAT ou Firewall ...), ICE passera par un serveur STUN pour obtenir une adresse réseau externe. Dans ce processus, les serveurs STUN n'ont qu'une seule tâche (Figure 19) : permettre à un pair derrière un NAT de découvrir son adresse publique et son port (Google fournit des serveurs STUN publics).

TURN n'est utilisé que pour relayer le flux data/audio et vidéo en streaming, mais pas les données de signalisation. Si l'on envisage de déployer un nombre important de clients, le

cas où la partie TURN doit relayer quelques dizaines ou centaines de communications vidéo et audio risque de poser un problème de charge. C'est ici que sans doute des passerelles ou des solutions commerciales seront nécessaires [ST, 18].

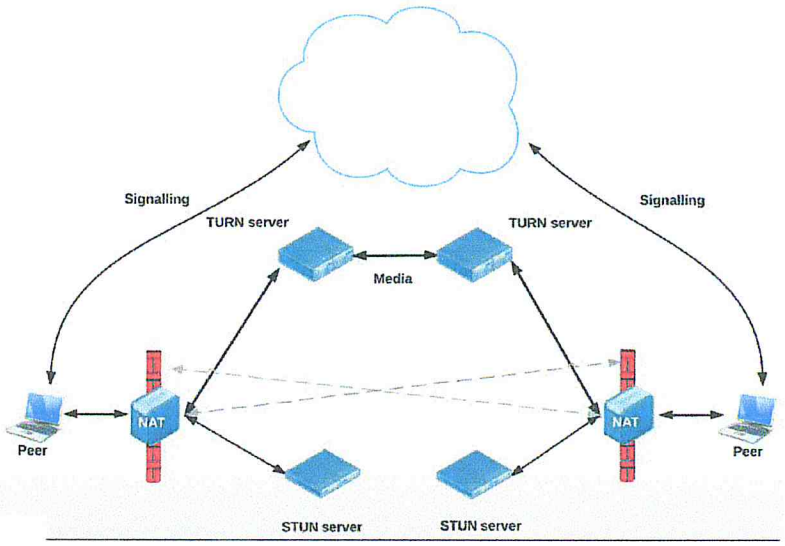


Figure 19 : Utilisation de serveurs STUN et TURN [ST,18].

## 7. Conclusion

Dans ce chapitre, nous avons présenté la technologie « Web Real Time Communication (WebRTC) » ayant donné récemment naissance à des innovations permettant la communication en temps réel grâce à un ensemble de technologies web temps réel. Nous avons également présenté les API nécessaires à la mise en œuvre d'une application utilisant cette technologie.

Dans le prochain chapitre, nous allons enchaîner avec la conception de la solution proposée exploitant la technologie WebRTC pour le contrôle télérobotique via Internet.

# Chapitre 3
# Conception du système de contrôle télérobotique

# 1. Introduction

Dans le but de développer une application web qui répond aux besoins des opérateurs, l'étape conceptuelle est considérée comme l'étape la plus importante et la plus sensible qui détermine l'utilité et l'utilisabilité de l'application logicielle. Elle requiert une transposition adéquate du monde réel à une solution informatique.

Tout au long de cette phase de conception, nous allons recourir au langage UML à travers ses différents diagrammes (diagramme de cas d'utilisation, diagramme de séquence, diagramme de classe et diagramme de déploiement) pour la modélisation de l'application web.

# 2. Unified Modeling Language (UML)

## 2.1. Définition

UML « Unified Modeling Language » ou « Langage de Modélisation Unifié » est un langage visuel constitué d'un ensemble de diagrammes ; chacun donne une vision différente du projet à traiter. UML fournit des diagrammes pour représenter le logiciel à développer : son fonctionnement, sa mise en route, les actions susceptibles d'être effectuées par le logiciel, etc. [Roques, 08].

## 2.2. Diagrammes UML

UML s'articule autour de quatorze types de diagrammes ; chacun d'entre eux étant dédié à la représentation des concepts particuliers d'un système logiciel. Ces types de diagrammes sont répartis en deux grands groupes [Roques, 08].

### 2.2.1. Diagrammes structurels

Ce type comporte 6 diagrammes différents : diagramme de classes, diagramme d'objets, diagramme de composants, diagramme déploiement, diagramme paquetage, et diagramme de structures composites.

### 2.2.2. Diagrammes comportementaux

Ce type se définit par 8 diagrammes différents ; chacun a un rôle bien défini : diagramme de cas d'utilisation, diagramme d'activités, diagramme d'états-transitions, diagrammes d'interaction, diagramme de séquence, diagramme de communication, diagramme global d'interaction, et diagramme de temps.

## 3. Web Application Extension

### 3.1. Définition

Web Application Extension (WAE) pour UML est une extension du langage UML pour la modélisation d'applications web. Il permet de représenter des pages Web et d'autres éléments architecturalement significatifs dans le but d'exprimer avec précision la totalité du système dans un modèle [Con, 02].

Cette extension UML est exprimée en termes de stéréotypes, de valeurs marquées, et de contraintes. Ces mécanismes permettent d'étendre la notation UML, ce qui permet de créer de nouveaux types de blocs de construction [Con, 02] :

- **Stéréotype** : c'est une extension du vocabulaire de la langue, permettant de joindre une nouvelle signification sémantique à un élément de modèle. Les stéréotypes peuvent être appliqués à presque tous les éléments du modèle et sont généralement représentés comme une chaîne entre une paire de guillemets : « ».
- **Valeurs associées** : c'est la définition d'une nouvelle propriété qui peut être associée à un élément du modèle. La plupart des éléments du modèle ont des propriétés qui sont associées.
- **Contrainte** : c'est une règle qui définit la manière dont le modèle peut être mis en place.

### 3.2. Différents types de stéréotype

WAE définit trois stéréotypes de classe de base et divers stéréotypes d'association [Con, 02] :

- **Classe :**
  - ***Server page*** : elle représente une page web dynamique qui possède des scripts exécutés par le serveur. Ces scripts interagissent avec des ressources du serveur, telles que les bases de données.
  - ***Opérations de l'objet*** : elles représentent les fonctions dans le script ; et ses attributs représentent les variables qui sont accessibles par les fonctions de la page.
  - ***Client page*** : elle représente une page web formatée en HTML (un mélange de données, de représentation et même de logique). Les fonctions d'une page client correspondent aux fonctions dans le script et ses attributs représentent les variables qui sont accessibles par les fonctions de la page.
  - ***HTML form*** : c'est un ensemble de champs de saisie faisant partie d'une page

client. La classe formulaire correspond à une balise HTML « Form » ; ses attributs sont les éléments de saisie telle qu'une zone de saisie, une zone de texte, un bouton d'option, etc.

- Un formulaire n'a pas d'opérations puisqu'il ne peut pas les encapsuler. Toute opération qui interagit avec le formulaire appartient à la page qui le contient.

- **Les associations :**
  - « **link** » : c'est une relation entre une page client et une autre page client ou une ressource du serveur. La cible peut être une classe *« client Page »* ou bien une classe *« server Page »*. Elle représente un pointeur entre ces pages ; elle peut représenter aussi une abstraction de l'élément d'ancrage HTML.
  - « **submit** » : c'est une relation directionnelle entre un formulaire et une page serveur. Les valeurs des champs du formulaire sont envoyées au serveur qui les traite par l'intermédiaire de pages serveur.
  - « **build** » : c'est une relation directionnelle entre les pages client et les pages serveur. Elle indique quelle page serveur est responsable de la création de la page client. Une page serveur peut construire plusieurs pages client ; mais, une page client n'est construite que par une et une seule page serveur.
  - « **redirect** » : c'est une relation directionnelle qui relie deux pages client ou serveur.
  - « **include** » : c'est une association directionnelle à partir d'une page serveur à une autre page serveur ou client ; elle indique que la page inclue est traitée.
  - « **forward** » **:** Une relation directionnelle entre une page serveur et une autre page serveur ou client. Cette association représente la délégation de traiter la demande d'un client pour une ressource à une autre page côté serveur.

- **Les attributs** :
  - « **input** » : il correspond à la balise <input> d'un formulaire HTML. Cet attribut est utilisé pour un mot ou une zone de texte.
  - « **select element** » : il permet à l'utilisateur de sélectionner une ou plusieurs valeurs dans une liste d'options.
  - « **textarea element** » : il correspond à la balise HTML <textarea> ; cet attribut permet à l'utilisateur de saisir un texte sur plusieurs lignes.

# 4. Processus de développement

## 4.1. Définition

Un processus définit une séquence d'étapes, partiellement ordonnées, qui concourent à l'obtention d'un système logiciel ou à l'évolution d'un système existant. L'objectif d'un processus de développement est de produire des logiciels de qualité qui répondent aux besoins de leurs utilisateurs dans des temps et des coûts prévisibles. Plus simplement, un processus doit permettre de répondre à la question fondamentale « *Qui fait quoi et quand ?* » [Roques, 08].

## 4.2. Processus unifié

### 4.2.1. Définition

UP « Unified Process » ou « Processus Unifié » est un processus de développement logiciel « *itératif et incrémental, centré sur l'architecture, conduit par les cas d'utilisation et piloté par les risques* » [Roques, 08]:

- *Itératif et incrémental* : le projet est découpé en itérations de courte durée (environ 1 mois) qui aident à mieux suivre l'avancement global. À la fin de chaque itération, une partie exécutable du système final est produite de façon incrémentale.
- *Centré sur l'architecture* : tout système complexe doit être décomposé en parties modulaires afin de garantir une maintenance et une évolution facile. Cette architecture (fonctionnelle, logique, matérielle, etc.) doit être modélisée en UML et pas seulement documentée en texte.
- *Piloté par les risques* : les risques majeurs du projet doivent être identifiés au plus tôt, mais surtout levés le plus rapidement possible. Les mesures à prendre dans ce cadre déterminent l'ordre des itérations.
- *Conduit par les cas d'utilisation* : le projet est mené en tenant compte des besoins et des exigences des utilisateurs, les cas d'utilisation du futur système sont identifiés, décrits avec précision et priorisés.

### 4.2.2. Phases du processus UP

La gestion d'un tel processus est organisée suivant les quatre phases suivantes : *initialisation*, *élaboration*, *construction* et *transition* [Roques, 08]:

- *Phase d'initialisation* : elle conduit à définir la « vision » du projet, sa portée, sa faisabilité, son business case, afin de pouvoir décider au mieux de sa poursuite ou de son arrêt.

- *Phase d'élaboration* : elle poursuit trois objectifs principaux en parallèle :
  - Identifier et décrire la majeure partie des besoins des utilisateurs.
  - Construire (et pas seulement décrire dans un document) l'architecture de base du système.
  - Lever les risques majeurs du projet.
- *Phase de construction* : elle consiste surtout à concevoir et implémenter l'ensemble des éléments opérationnels (autres que ceux de l'architecture de base). C'est la phase la plus consommatrice en ressources et en effort.
- *Phase de transition* : elle permet de faire passer le système informatique des mains des développeurs à celles des utilisateurs finaux.

## 5. Conception de l'application

Dans le cadre de ce travail, nous allons développer une plateforme de contrôle à distance de robots en utilisant des technologies récentes dites web temps réel (WebRTC). Nous nous limitons aux fonctionnalités essentielles permettant de mener à bien un travail de contrôle ; en particulier, celles liées au contrôle de robots (base mobile, bras manipulateur, manipulateur mobile, caméra, etc.).

Dans ce qui suit, nous présentons la partie conceptuelle de notre « plateforme de contrôle à distance de robots ». Nous commençons par les diagrammes de cas d'utilisation qui représentent les fonctionnalités du système, suivis par des diagrammes de séquence ; par la suite, nous nous intéressons à l'aspect structurel avec les diagrammes de classe et le diagramme de déploiement, et à l'aspect dynamique avec les diagrammes de séquence.

### 5.1. Diagrammes de cas d'utilisation

Le diagramme de cas d'utilisation permet une description du système à construire en prenant le point de vue de l'utilisateur. Les deux concepts de base de ce diagramme sont l'acteur et le cas d'utilisation [Roques, 08].

#### 5.1.1. Acteurs

Un acteur représente un ensemble cohérent de rôles joués par des entités externes (opérateur humain, matériel ou autre système) qui interagissent directement avec le système étudié par échange de messages et pouvant consulter ou modifier directement l'état du système. Il existe deux catégories d'acteurs :

- *Acteurs principaux* : ce sont des entités qui sollicitent le système pour réaliser une tâche bien précise.
- *Acteurs secondaires* : ce sont des entités qui sont sollicitées par le système lors de la réalisation d'une tâche.

Pour le cas de notre application web, nous avons identifié trois acteurs ; ils sont donnés comme suit :

- *Administrateur* : c'est l'administrateur de la plateforme.
- *Utilisateur* : c'est un internaute qui est déjà inscrit (possédant un compte) et qui s'est authentifié.
- *Visiteur* : c'est un internaute qui ne s'est pas encore authentifié.

### 5.1.2. Diagramme de cas d'utilisation général

Le diagramme de cas d'utilisation général, donné par la figure 20, regroupe tous les cas d'utilisation de base afin d'avoir une vue globale du fonctionnement du système et de mettre en évidence les éventuelles relations entre les cas d'utilisation.

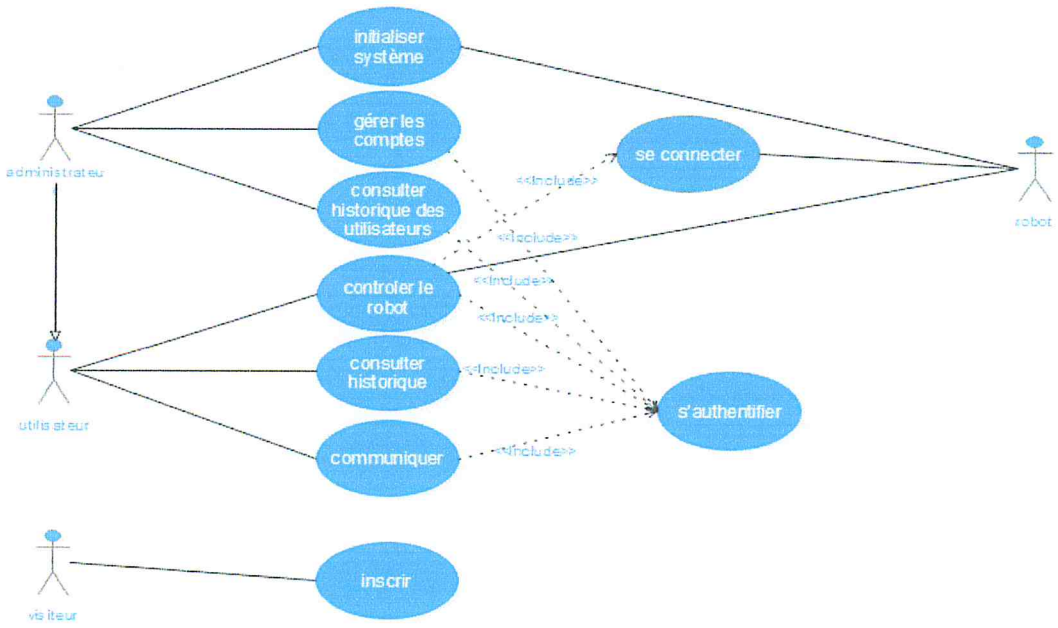


Figure 20 : Diagramme de cas d'utilisation global.

### 5.1.3. Diagramme de cas d'utilisation « Contrôler le robot »

Le contrôle du robot est effectué par les différents utilisateurs qui peuvent contrôler le bras manipulateur, la base mobile, et la caméra (Figure 21).

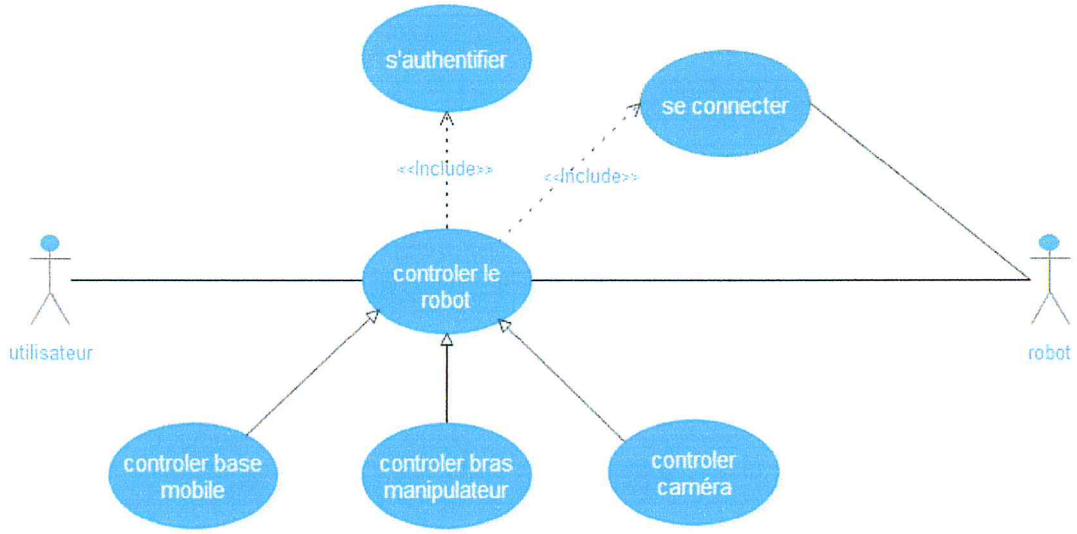


Figure 21 : Diagramme de cas d'utilisation « Contrôler le robot ».

### 5.1.4. Diagramme de cas d'utilisation « Gérer les comptes »

La gestion des comptes est effectuée par l'administrateur qui peut effectuer une mise à jour, valider ou supprimer un compte, etc. (Figure 22).

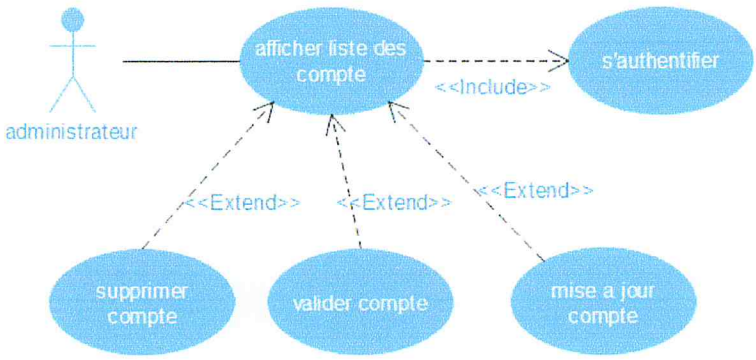


Figure 22 : Diagramme de cas d'utilisation « Gérer les comptes ».

## 5.2. Diagrammes de séquence

Les diagrammes de séquence présentent la vue dynamique du système. Leur objectif est de représenter les interactions entre les objets en indiquant la chronologie des échanges. Cette représentation est réalisée par des cas d'utilisation [Roques, 08].

### 5.2.1. Cas d'utilisation « Authentification »

Ce diagramme représente l'interaction entre l'utilisateur et le système afin de se connecter à la plateforme (Figure 23).

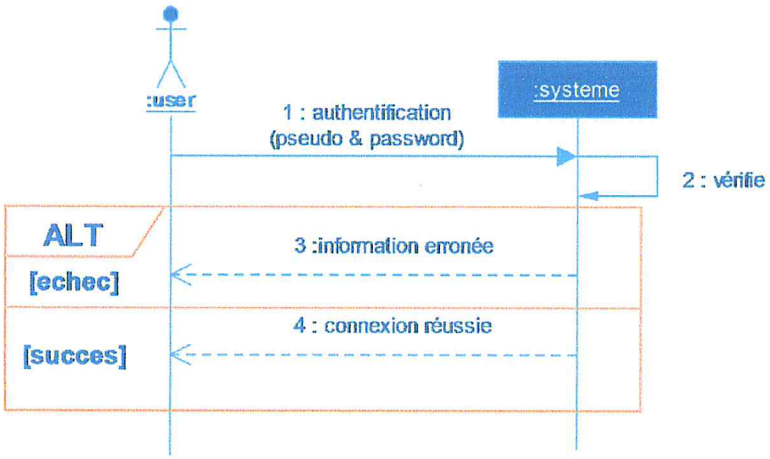


Figure 23 : Diagramme de séquence système « Authentification ».

### 5.2.2. Cas d'utilisation « Connecter »

Ce diagramme montre comment l'utilisateur et le robot peuvent établir une connexion directe entre eux via WebRTC (Figure 24).

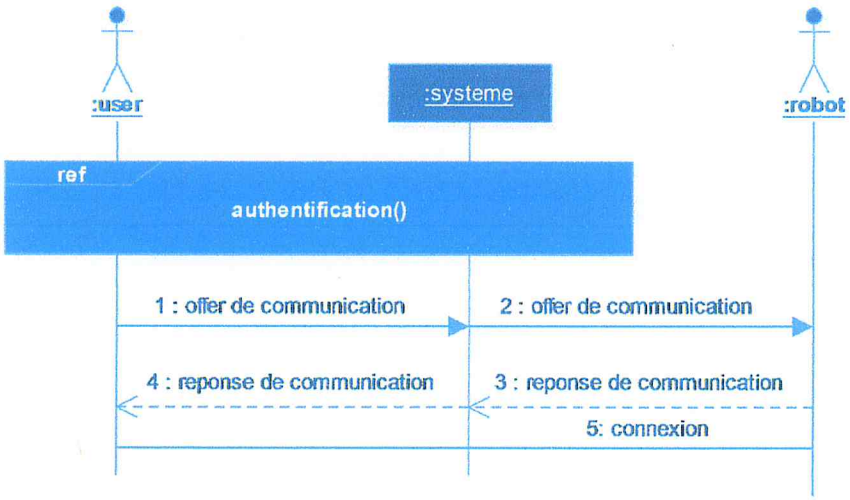


Figure 24 : Diagramme de séquence système « Connecter ».

### 5.2.3. Cas d'utilisation « Contrôler le bras manipulateur »

Ce diagramme de cas d'utilisation explique comment un utilisateur peut manipuler le robot manipulateur (Figure 25).

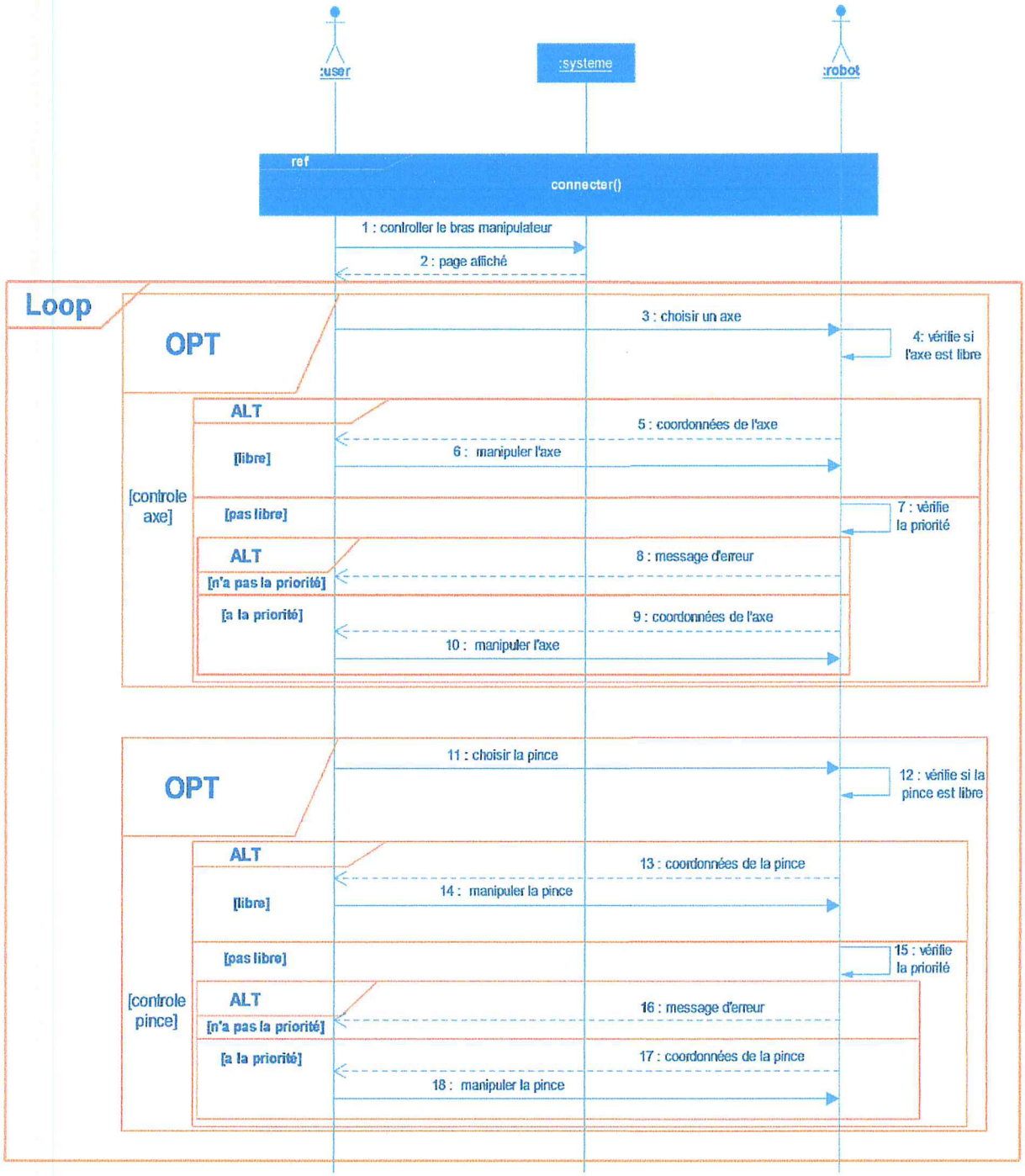


Figure 25 : Diagramme de séquence système « Contrôler le bras manipulateur ».

### 5.2.4. Cas d'utilisation « Contrôler la base mobile »

Ce diagramme illustre les interactions entre l'utilisateur, le système et le robot pour permettre à l'utilisateur de contrôler la base mobile du robot (Figure 26).

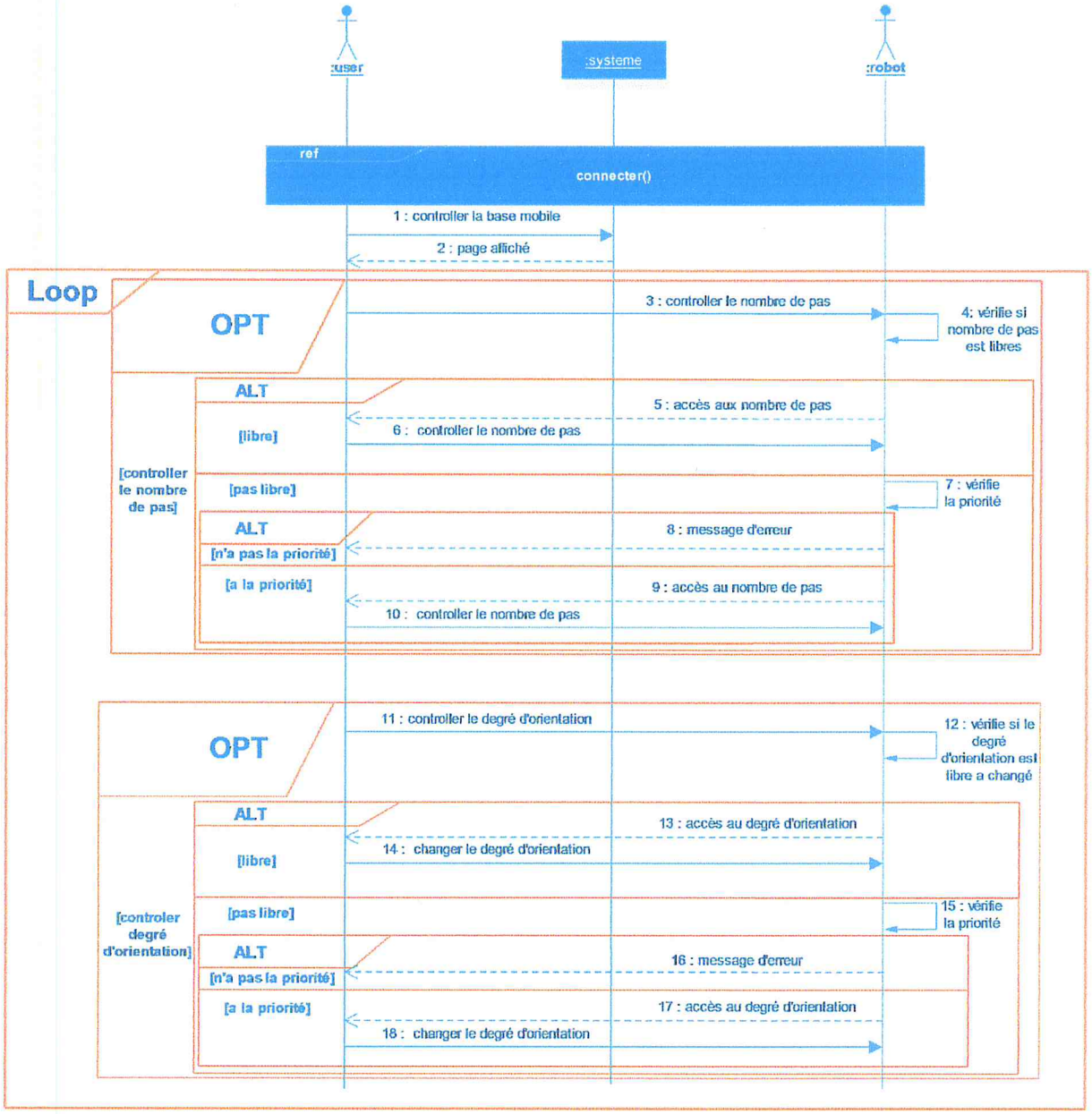


Figure 26 : Diagramme de séquence système « Contrôler la base mobile ».

### 5.2.5. Cas d'utilisation « Commander la caméra »

Ce diagramme de cas d'utilisation (Figure 27) illustre l'ensemble des commandes de la caméra.

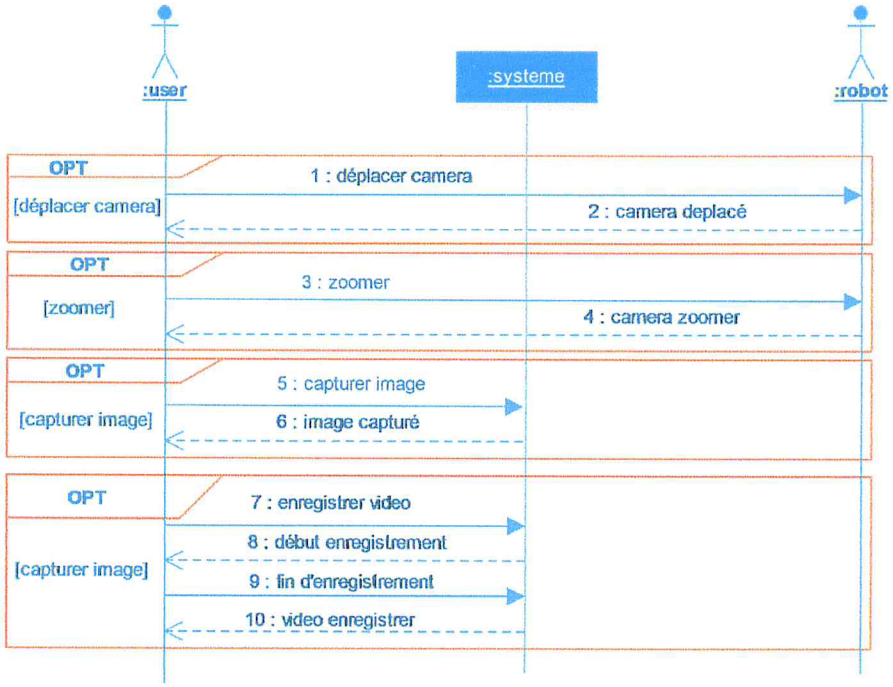


Figure 27 : Diagramme de séquence système « Commander la caméra »

### 5.2.6. Cas d'utilisation « caméra »

Ce diagramme (Figure 28), explique comment un « user » peut contrôler une « caméra » installée sur le robot distant.

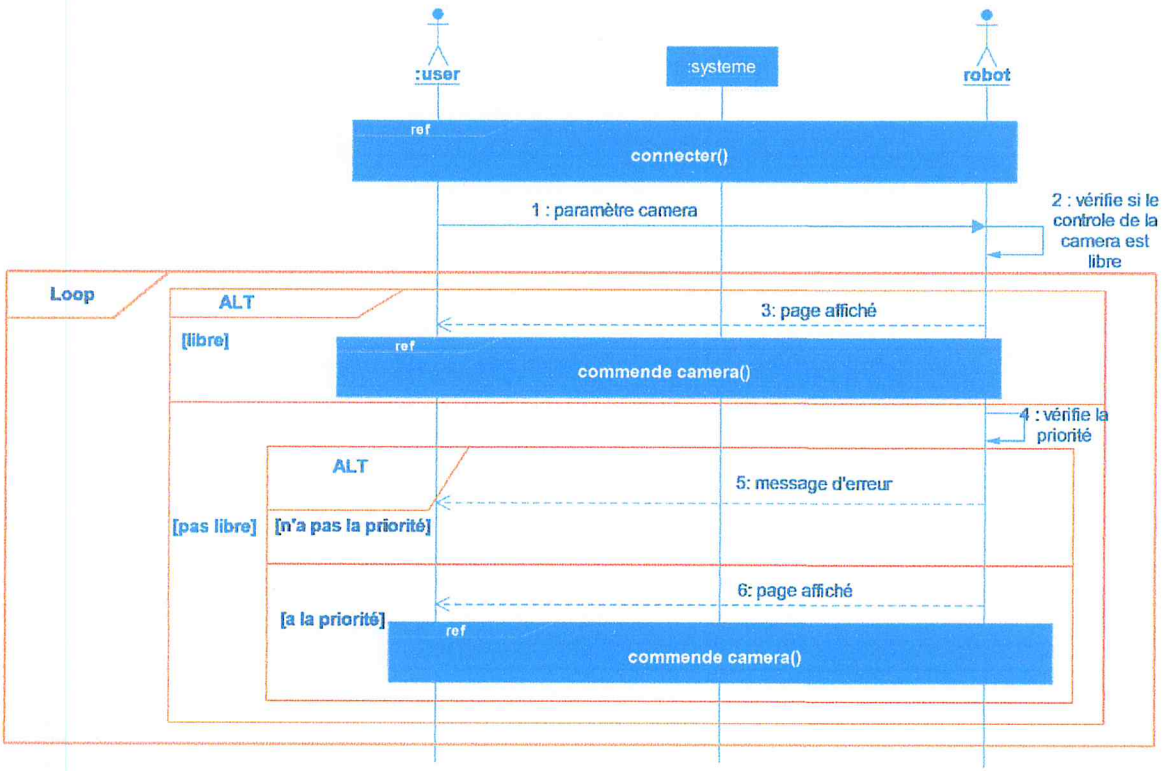


Figure 28 : Diagramme de séquence système « caméra ».

## 5.3. Diagramme de classes

Le diagramme de classes (Figure 29) est un diagramme structurel ne présentant que les classes (pas les instances de classes). Il permet de décrire la structure interne des classes en termes d'attributs et d'opérations. Il permet aussi de représenter les associations statiques entre les classes ; cependant, ce diagramme ne décrit pas comment les liens effectifs entre les instances sont effectués [Roques, 08].

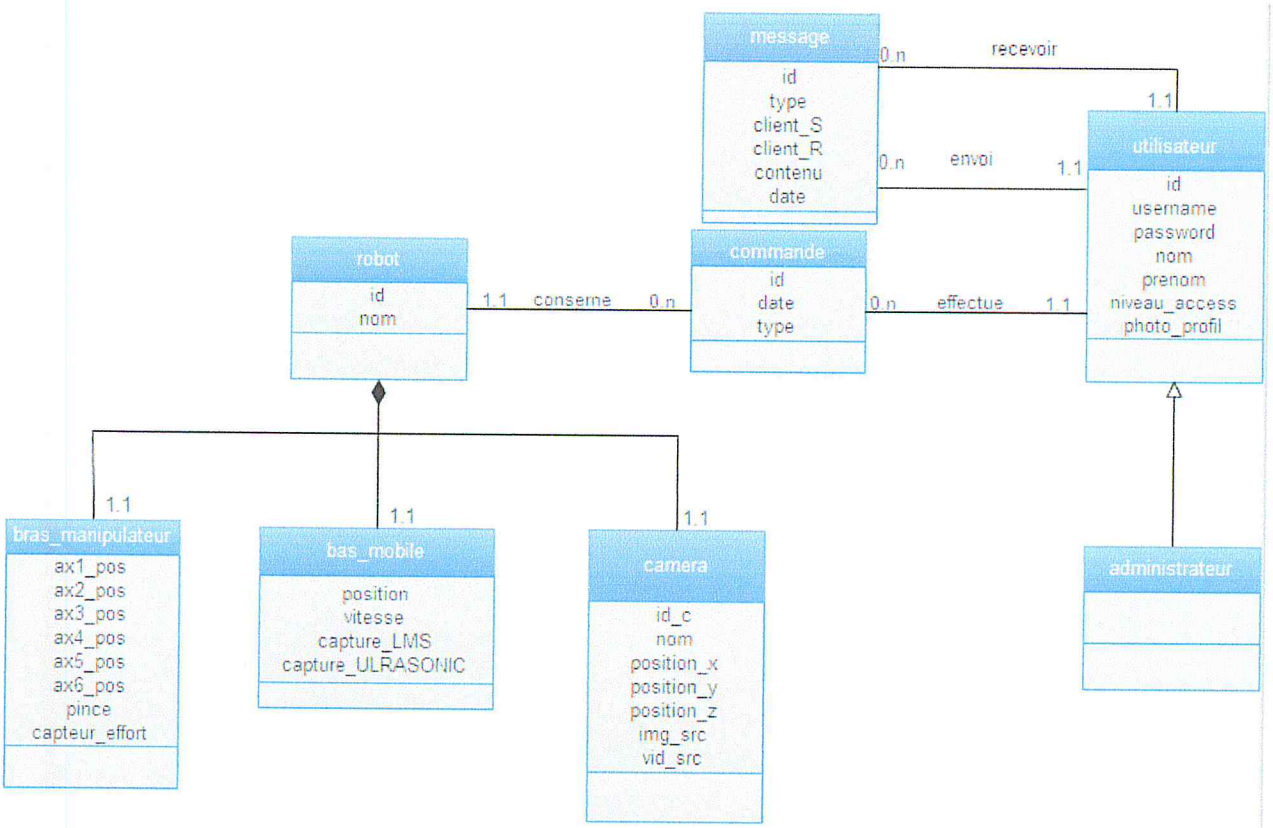


Figure 29 : Diagramme de classes.

## 5.4. Diagramme de classes en utilisant WAE

La Figure 30 représente un diagramme de classes qui décrit, en utilisant WAE, les classes les plus importantes utilisées dans le cadre de ce travail sous la forme de pages. Les relations entre ces différents éléments sont également indiquées sur le diagramme.

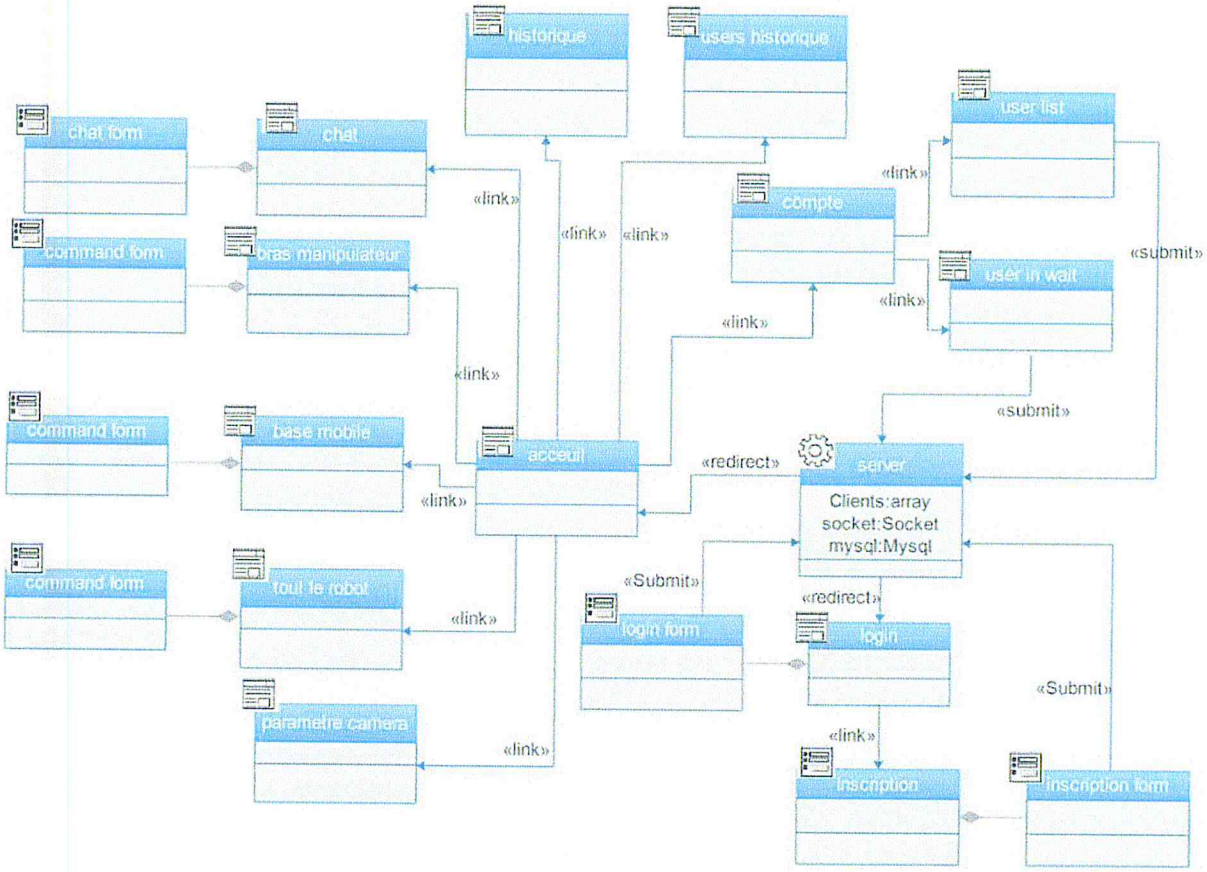


Figure 30 : Diagramme de classes en utilisant WAE

## 5.5. Diagrammes de séquence

### 5.5.1. Diagramme de séquence « Authentification »

Ce diagramme de séquence décrit de manière détaillée l'authentification d'un utilisateur (Figure 31).

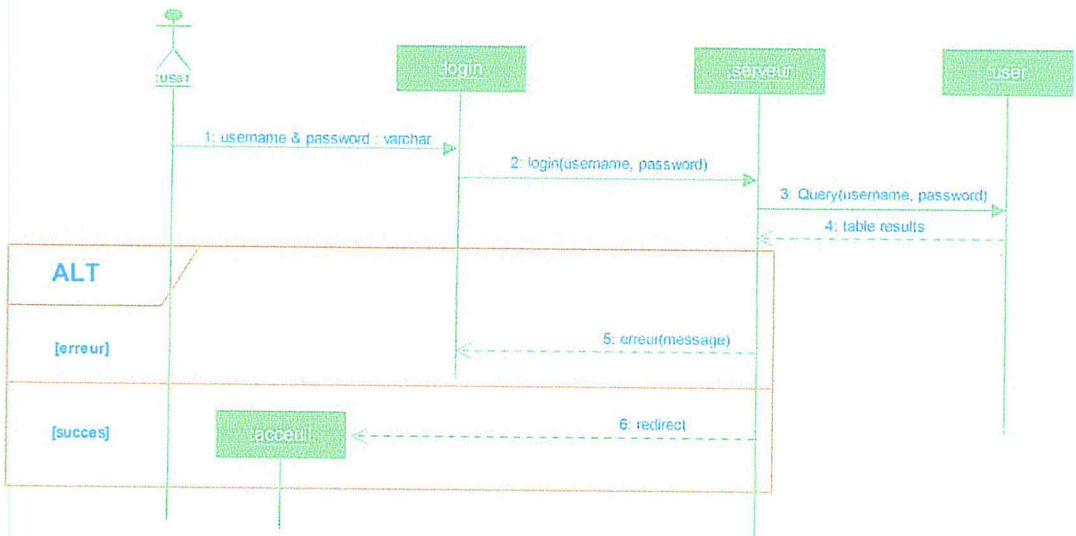


Figure 31 : Diagramme de séquence « Authentification ».

### 5.5.2. Diagramme de séquence « Connecter »

Ce diagramme définit dans un ordre chronologique les séquences d'événements nécessaires pour établir une connexion WebRTC entre le client et le robot (Figure 32).

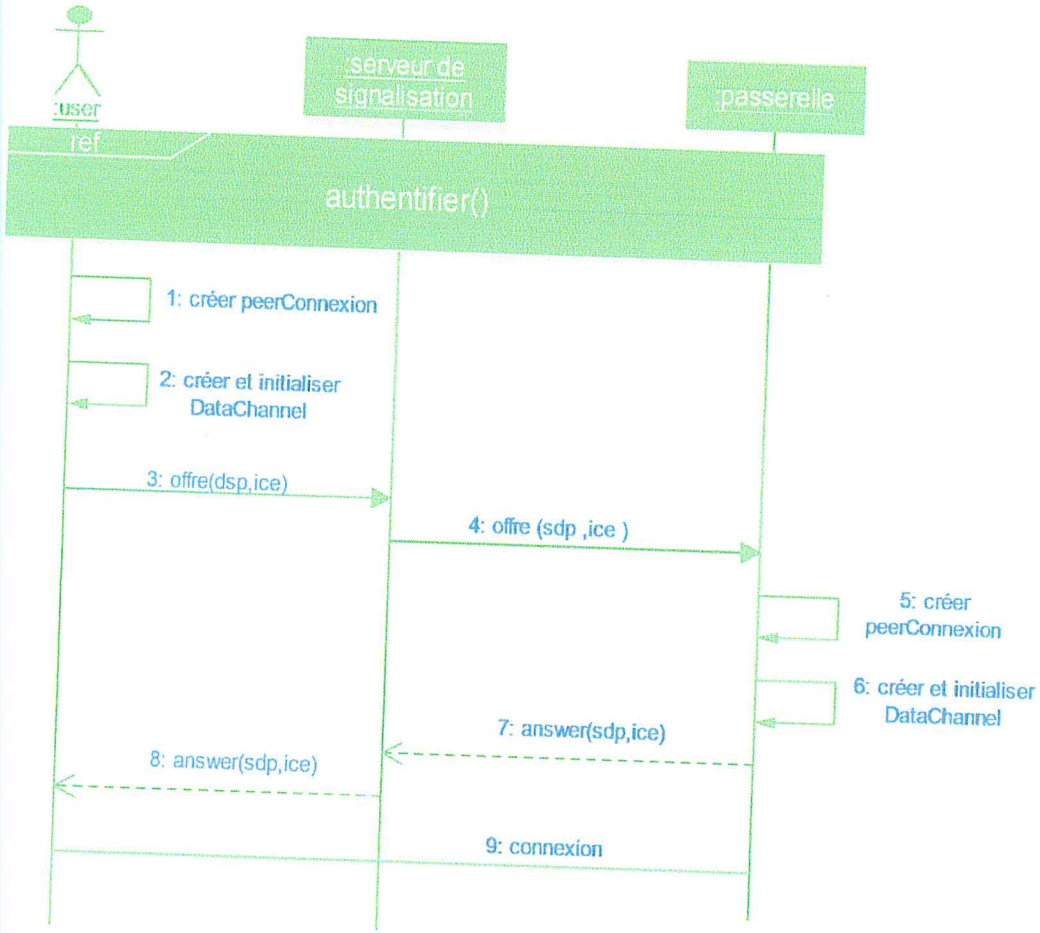


Figure 32 : Diagramme de séquence « Connecter ».

### 5.5.3. Diagramme de séquence « Contrôler un axe »

Ce diagramme représente le processus de contrôle d'un axe du bras manipulateur par un utilisateur (Figure 33).

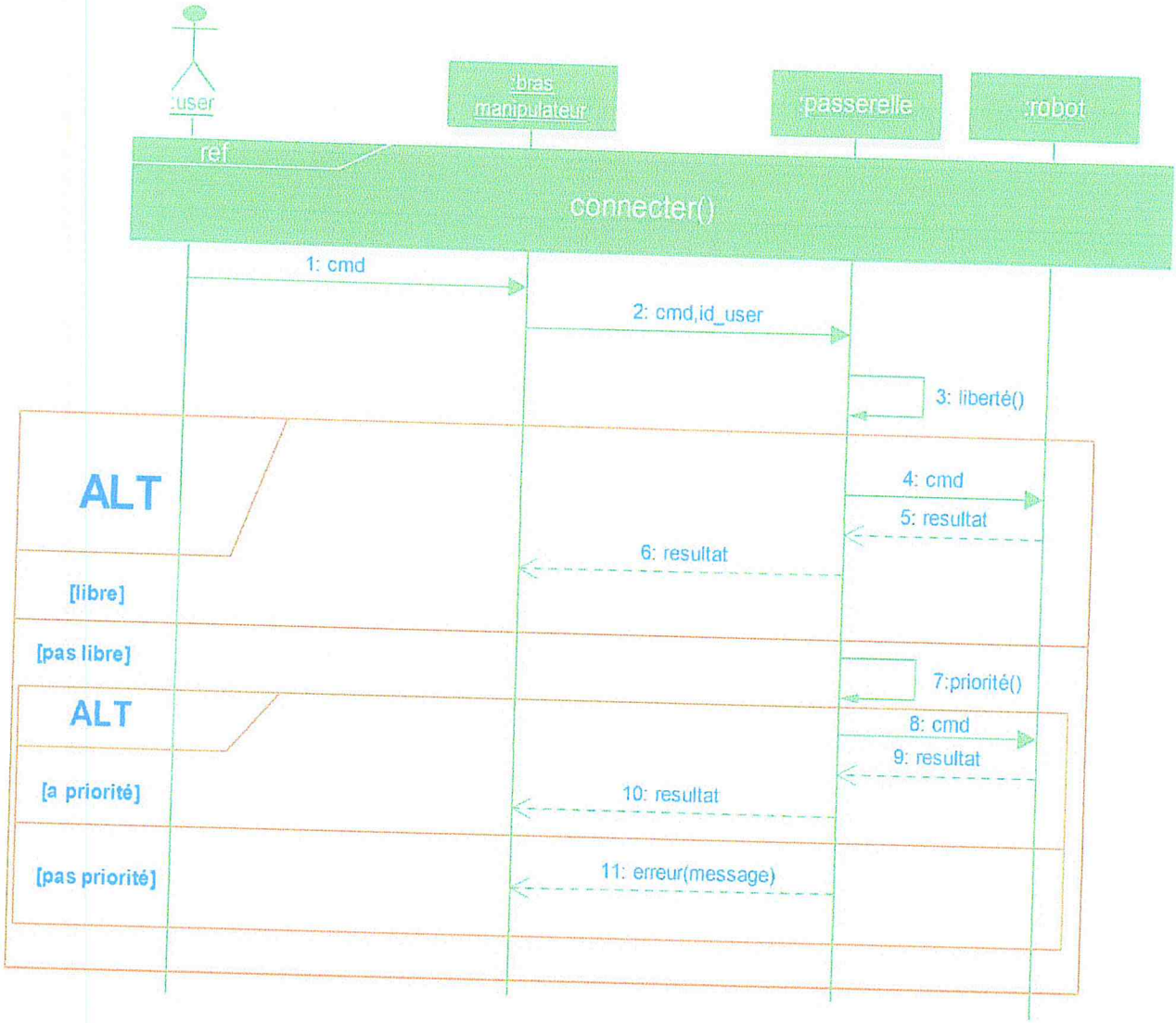


Figure 33 : Diagramme de séquence « Contrôler un axe ».

### 5.5.4. Diagramme de séquence « Contrôler l'orientation de la base mobile »

Ce diagramme illustre, étape par étape, le processus de contrôle de l'orientation de la base mobile par un utilisateur (Figure 34).

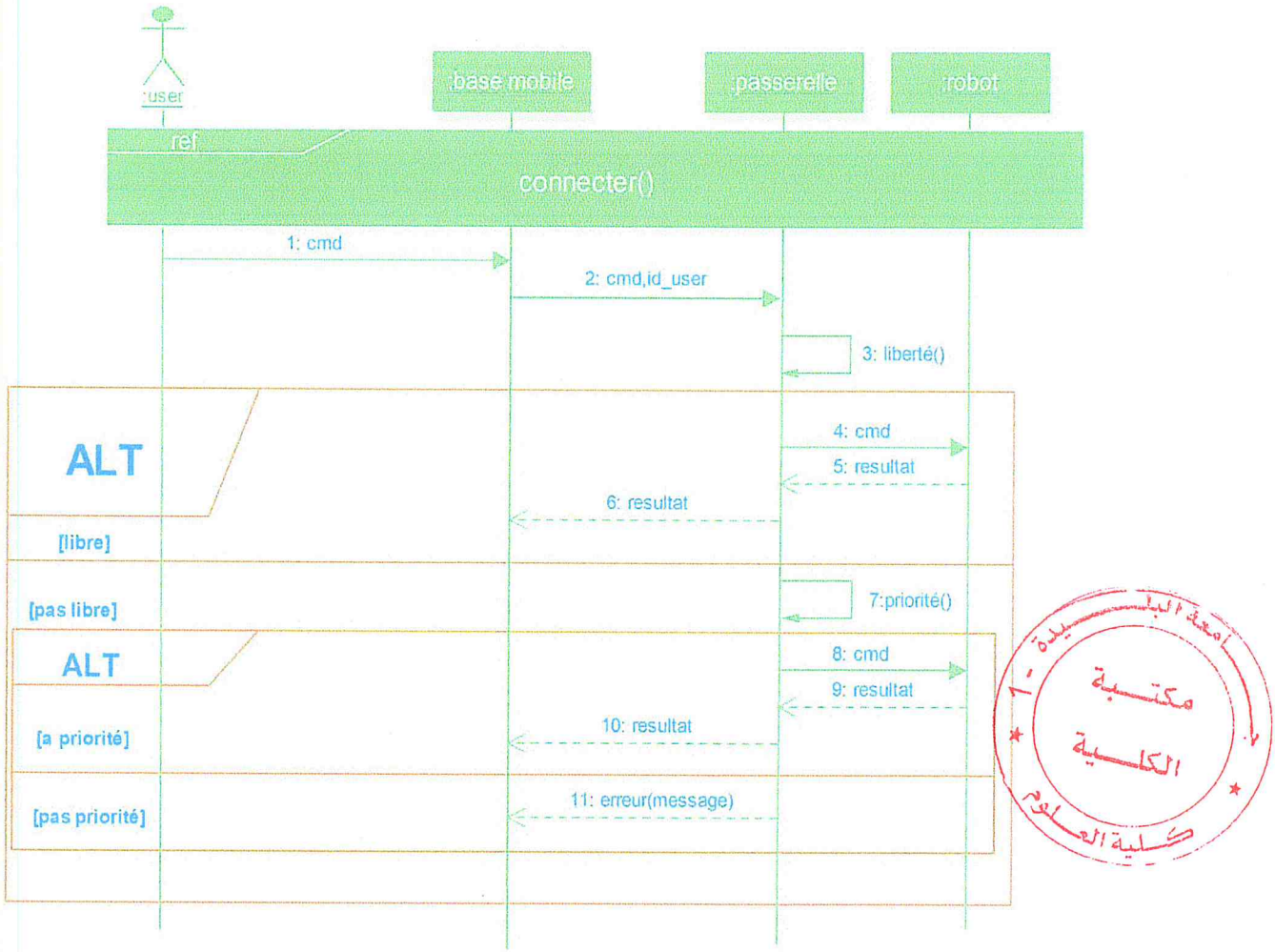




Figure 34 : Diagramme de séquence « Contrôler l'orientation de la base mobile ».

### 5.5.5. Diagramme de séquence « Zoomer la caméra »

Ce diagramme, donné par la Figure 35, présente en détail le processus de zoom la caméra par l'utilisateur.

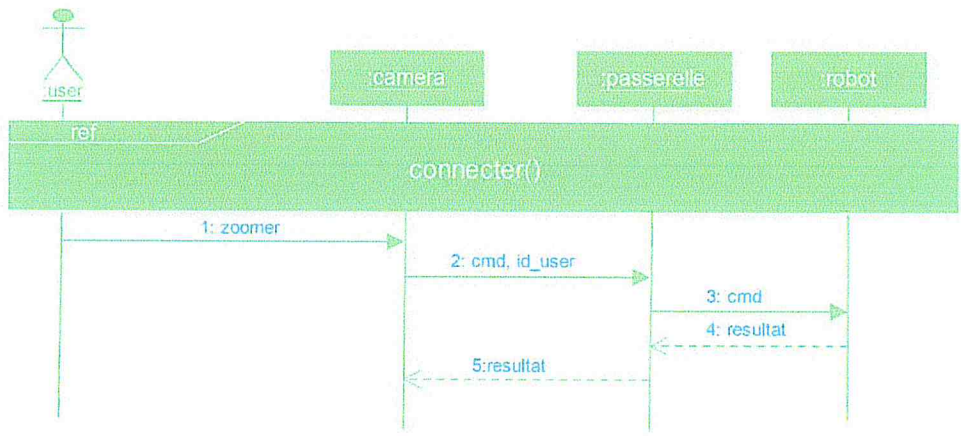


Figure 35 : Diagramme de séquence « Zoomer la caméra ».

### 5.6. Diagrammes de déploiement

Un diagramme de déploiement décrit la disposition physique des ressources matérielles qui composent le système. De plus, ce diagramme montre la répartition des composants sur ces matériels. Chaque ressource étant matérialisée par un nœud ; le diagramme de déploiement précise comment les composants sont répartis sur les nœuds et quelles sont les connexions entre les composants ou les nœuds [Diplio, 2000].

La figure 36 illustre le diagramme de déploiement de notre système.

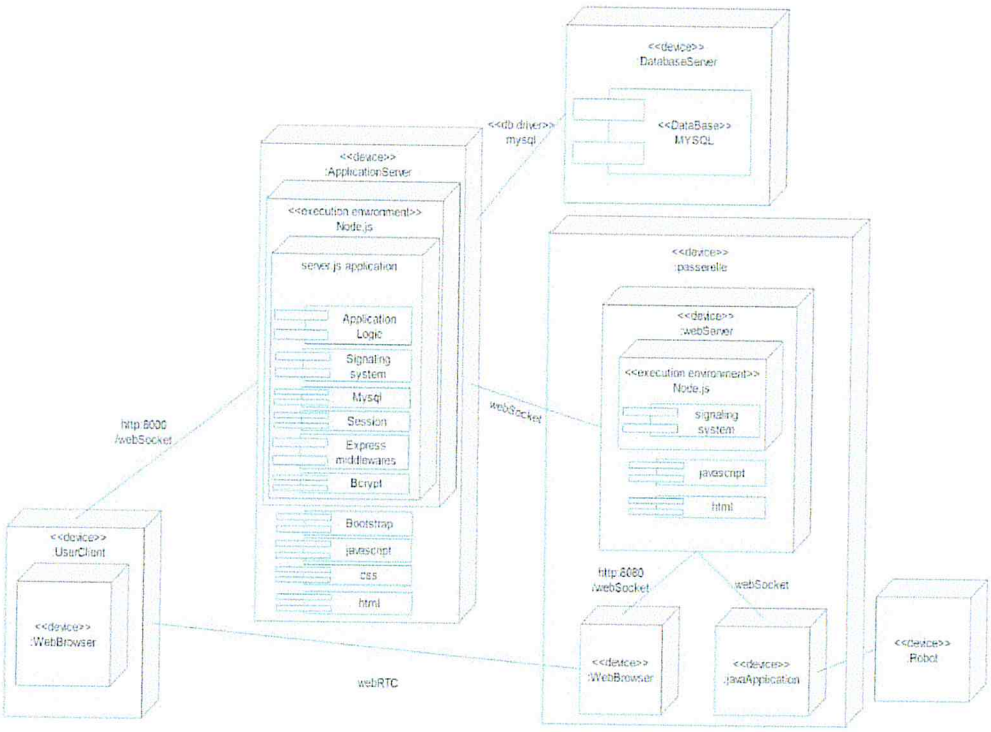


Figure 36 : Diagramme de déploiement.

## 6. Conclusion

La modélisation du système avant sa réalisation permet de mieux connaître le fonctionnement du système, de déceler les éventuels problèmes et permet aussi de mieux gérer la complexité de notre système. En outre, nous avons présenté le langage de modélisation unifié UML pour la modélisation du système ainsi que le processus unifié UP que nous avons suivi pour le développement de notre système.

Afin de bien illustrer la conception de notre système, nous avons utilisé plusieurs diagrammes tels que le diagramme de cas d'utilisation, le diagramme de classes, le diagramme de séquence ainsi que le diagramme de déploiement.

Le chapitre suivant présente l'implémentation de notre système de contrôle télérobotique exploitant la technologie WebRTC.

# Chapitre 4

# Implémentation du système

## 1. Introduction

Ce chapitre est consacré à l'implémentation et la mise en œuvre de notre système de contrôle télérobotique. Nous commençons par la description de l'architecture de notre application ainsi que les outils de développement utilisés. Ensuite, nous présentons les interfaces utilisateur du système. Enfin, nous terminons par la représentation de quelques tests expérimentaux et la discussion des principaux résultats obtenus.

## 2. Architecture de l'application

La Figure 37 montre l'architecture de l'application ainsi que les différents protocoles et technologies utilisés.

Notre application se déploie sur une architecture distribuée qui dispose d'un ensemble d'entités qui sont reliées par un réseau de communication. L'application est constituée d'un serveur, des clients, et une passerelle et d'un robot qui se situe dans un site distant. Les échanges s'effectuent par une connexion directe entre les pairs concernés, ce qui permet de gagner du temps lors de l'exécution de tâches.

Concernant la connexion entre les opérants et le robot, l'utilisateur se connecte en premier lieu au serveur d'application pour accéder à l'application web. Ensuite, il envoie une offre au serveur de signalisation, qui à son tour la renvoie à la passerelle. Dès que cette dernière reçoit l'offre d'un utilisateur, elle lui renvoie une réponse pour permettre l'ouverture du canal WebRTC pour une communication en temps réel entre les deux entités.

En ce qui concerne la passerelle, quand une commande est reçue par l'application web de la part d'un utilisateur, elle la renvoie au serveur local. Ce dernier renvoie cette commande à l'application exécutif pour procéder à son traitement et son envoi au robot.

Finalement, notre application repose sur l'utilisation de l'architecture client léger (l'application utilise HTML et JavaScript coté client, donc le client doit avoir un navigateur web assez récent).

En ce qui concerne les différents composants de notre application :

- *Serveur d'application* : dans ce cas, Node.js est utilisé pour la création du serveur web puisqu'il utilise un modèle d'entrée/sortie non-bloquant qui lui-même permet de gérer plusieurs tâches de façon asynchrone.
- *Pages web* : elles sont implémentées en HTML, CSS et JavaScript.

- *Communication* : la communication entre le client et le serveur est établie grâce à socket.io.
- *Passerelle* : elle est composée de trois parties (*i*) un serveur local (en Node.js), (*ii*) une application web, et (*iii*) une application exécutif (en java).
- *Communication temps réel* : la communication temps réel entre les pairs est établie grâce aux API du WebRTC.

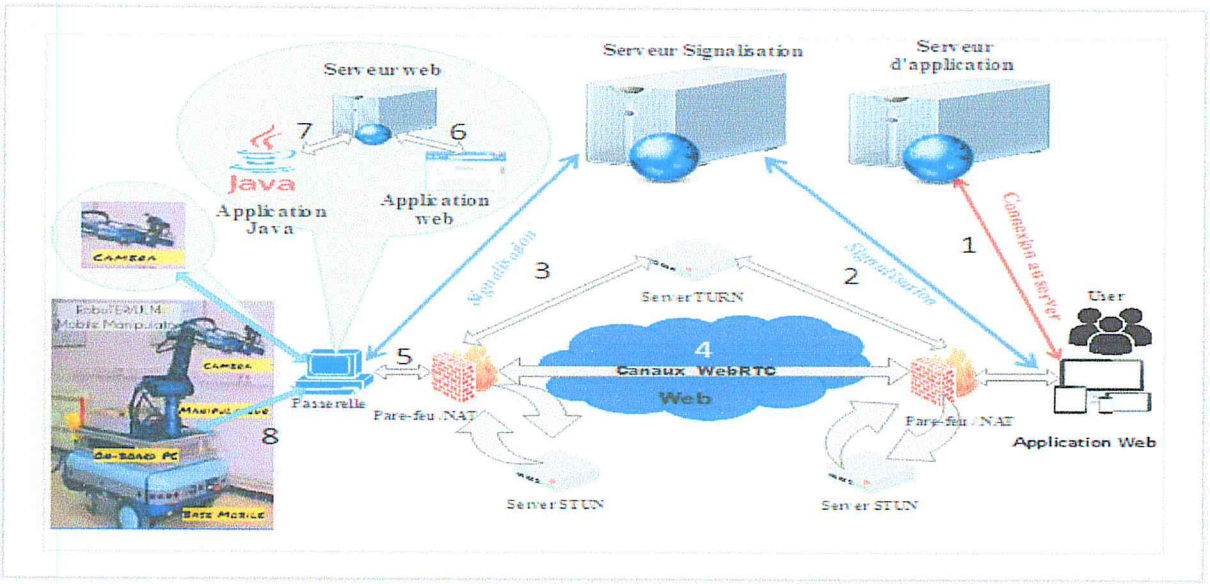


Figure 37 : Architecture de l'application.

## 3. Langages de développement

Notre projet vise à développer une application de contrôle télérobotique en temps réel qui utilise un ensemble de technologies web et des outils JavaScript. Pour créer un site web, HTML et CSS sont indispensables. Cependant, ces langages ne suffisent pas pour le réaliser ; en effet, il faut compléter ces deux langages par d'autres. Dans notre cas, nous avons utilisé le langage JavaScript car WebRTC et Node.js sont basés seulement sur ce langage de script, et aussi le langage de requêtes SQL qui permet de communiquer avec la base de données. Et à la fin, nous avons utilisé le langage java avec son aspect orienté objet qui permet de faciliter le traitement des commandes à envoyer au robot.

## 4. Environnement de développement

Lors du développement de notre système, nous avons utilisé les outils logiciels suivants :

- **Sublime text 3** : c'est un éditeur de texte et de code pour Windows. Il permet l'édition de plusieurs documents simultanément qui s'adapte aux langages de développement web. Cet éditeur offre de nombreuse facilités : coloriage adapté au langage, recherche avancée dans les dossiers, auto-complétion, simplicité d'utilisation, légèreté de démarrage et de manipulation, etc. [Sub, 13].
- **Interpréteur Cmder** : c'est une console configurée pour reconnaître Node.js. *Cmder* permet de lancer les applications Node.js (Figure 38) ; son interface permet d'exécuter plusieurs instances dans différents onglets, ce qui permet de travailler facilement dans plusieurs dossiers simultanément [Cmdr, 16].

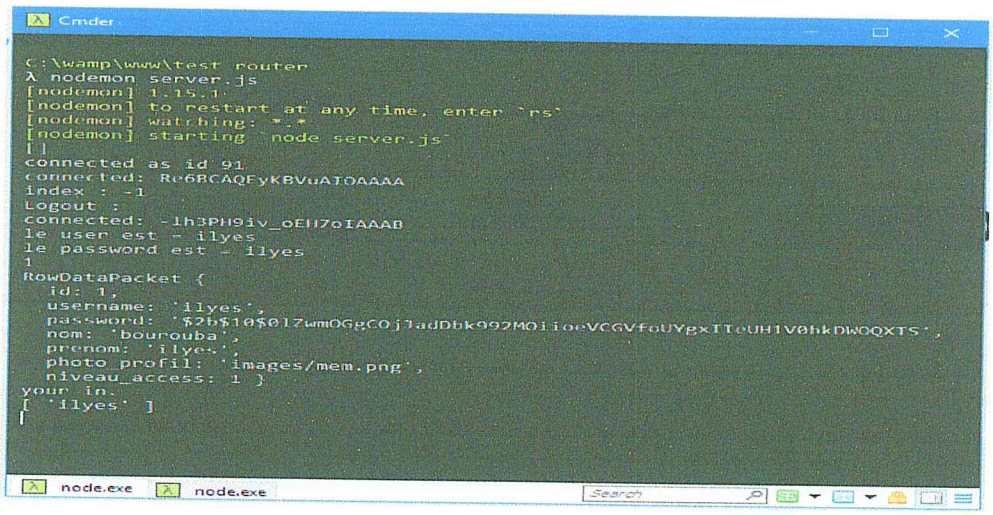


Figure 38 : Node.js - L'interpréteur Cmder.

- **NetBeans 8.2** : Dans notre projet, nous avons utilisé NetBeans 8.2 pour programmer en JAVA. C'est un environnement de développement intégré (IDE) pour Java, placé en open source par Sun en juin 2000 sous licence CDDL (Common Développement and Distribution License). En plus de Java, NetBeans supporte également différents langages, comme Python, C/C++, XML et HTML. Il comprend toutes les caractéristiques d'un IDE moderne (éditeur en couleur, projets multi-langages, refactoring, éditeur graphique d'interfaces et de pages web). NetBeans est disponible sous Windows, Linux, Solaris (sur x86 et SPARC), Mac OS X et Open VMS. Cependant, comme NetBeans est lui-même développé en Java, il est assez lent et gourmand en ressources mémoires [Netb, 16].

# 5. Implémentation

## 5.1. Partie serveur

### 5.1.1. Serveur Node.js

Node.js est une plateforme logicielle libre et événementielle en JavaScript pour construire facilement des applications de réseau rapides et évolutives. Elle utilise la machine virtuelle V8 de Google Chrome. Node.js contient une bibliothèque de serveur HTTP intégrée, ce qui rend possible de construire ou de créer un serveur web sans avoir besoin d'un logiciel externe comme Apache ou autres. Il permet aussi de mieux contrôler la façon dont le serveur web fonctionne, tout en étant idéal pour les applications en temps réel intensives en données qui courent à travers des dispositifs distribués.

#### 5.1.1.1. Modules NPM utilisés

Node.js contient plusieurs modules qui ont différentes fonctionnements ; mais, leur but est de faciliter l'utilisation de Node.js. Voici une liste des quelques modules utilisés dans notre application :

- **Express :** permet la création de l'application et de ses comportements.
- **Bcrypt :** c'est une fonction de hachage qui permet de crypter et de décrypter les mots de passe.
- **MySQL :** les interactions avec la ou les bases de données.
- **http :** créer le serveur HTTP.
- **Hbs :** un moteur de modèle pour faciliter l'écriture du code html.
- **Passport :** permet de gérer l'authentification grâce à son système de gestion de cookies et sessions.
- **socket.io :** pouvoir interagir en temps réel entre les clients et le serveur.
- **Socket.io-client :** permet l'interaction entre deux serveurs.

La Figure 39 est une partie du code JavaScript du serveur qui expose les différents modules utilisés dans notre plateforme de travail collaboratif.

```
var express = require('express');
var app = express();
var bodyParser = require('body-parser');
var server = require('http').createServer(app);
var io = require('socket.io')(server);
var session = require('express-session');
var MySQLStore = require('express-mysql-session')(session);
var passport = require('passport');
var mysql = require('mysql');
var bcrypt = require('bcrypt');
var hbs = require('hbs');
var stor = require('data-store')('my-app');
```

Figure 39 : Partie du code de serveur.js qui montre l'usage des modules.

### 5.1.1.2. Mise en place des comportements

La figure 40 ci-dessous donne une partie du code du serveur (fichier server.js) qui montre l'authentification d'un utilisateur à son compte. À cet effet, le système vérifie l'existence du pseudonyme et la correspondance du mot de passe saisi au niveau de la base de données. Dans le cas où ces données sont justes, le serveur le redirige vers la page d'accueil, dans le cas contraire, il retourne vers la page d'authentification accompagnée d'un message d'erreur.

```
app.post('/login', (req, res)=>{
    var user = req.body.username
    var pass = req.body.password
    console.log("le user est = " + user + " \nle password est = " + pass);
    connection.query('SELECT * FROM user WHERE username = ? ', [user], function (error, results, fields) {
        if (error) throw error;
        if (results.length==1){
            if (bcrypt.compareSync(pass, results[0].password)) {
                console.log(results.length);
                console.log(results[0]);
                console.log('your in.')

                users.push( results[0].username);
                console.log(users)

                const user_id =  results[0]

                req.login(user_id,(err)=>{
                    res.redirect('/home');
                })
            }else {
                console.log('wrong pass')
                check = false
                res.redirect('/');
            }
        }else {
            check = false
            res.redirect('/');
        }
    });
});
```

Figure 40 : Extrait du code server.js.

### 5.1.2. WampServer 3.1.0

WampServer (Windows Apache MySQL PHP) est une plateforme de développement d'applications Web dynamiques sous Windows. Il intègre un serveur Apache 2.4 (serveur HTTP), un serveur MySQL (un SGBD : Système de Gestion de Base de Données), un

interpréteur PHP, ainsi qu'une interface Web PhpMyAdmin permettant la gestion et l'administration de base de données [Wamp, 18].

Dans notre cas, nous avons utilisé WampServer 3.1.0 [Wamp, 18] qui intègre MySQL 5.7.19, PHP 5.6.31 et PHPMyAdmin 4.7.4.

## 5.2. Partie client

### 5.2.1. Pages HTML et code JavaScript

Les pages HTML qui nécessitent l'utilisation des modules doivent récupérer les fichiers JavaScript qui permettent d'effectuer des actions du côté du client pour communiquer avec le serveur.

Pour le développement des pages web, nous avons utilisé Bootstrap [Bot,12]. Nous avons constaté que Bootstrap apporte un gain non négligeable de facilité et de rapidité d'intégration.

### 5.2.2. Bootstrap

Bootstrap est un framework très récent qui fait partie des plus populaires. C'est un framework front-end gratuit pour le développement web. Il contient plusieurs outils utiles à la réalisation de sites interactifs. Il permet, entre autres, de concevoir plus facilement un design responsive qui va permettre d'adapter l'affichage de l'application à tout type d'écran. Il a été conçu en 2010 par deux développeurs de chez Twitter. Son but était de diminuer les coûts de maintenance dus aux incohérences entre les différentes bibliothèques existantes [Bot,12].

La Figure 41 montre comment se fait la récupération des fichiers JavaScript du côté du client comme socket.io.js. Celui-ci est automatiquement fourni par le serveur Node.js via le module socket.io.

```
<script src="bootstrap/jquery/dist/jquery.min.js"></script>
<!-- Bootstrap -->
<script src="bootstrap/js/bootstrap.min.js"></script>
<!-- FastClick -->
<script src="bootstrap/fastclick/lib/fastclick.js"></script>
<!-- NProgress -->
<script src="bootstrap/nprogress/nprogress.js"></script>
<!-- Chart.js -->
<script src="bootstrap/Chart.js/dist/Chart.min.js"></script>
<!-- gauge.js -->
<script src="bootstrap/bernii/gauge.js/dist/gauge.min.js"></script>
<!-- bootstrap-progressbar -->
<script src="bootstrap/bootstrap-progressbar/bootstrap-progressbar.min.js"></script>
<!-- iCheck -->
<script src="bootstrap/iCheck/icheck.min.js"></script>
<!-- Skycons -->
<script src="bootstrap/skycons/skycons.js"></script>
<!-- Flot -->
<script src="bootstrap/Flot/jquery.flot.js"></script>
<script src="bootstrap/Flot/jquery.flot.pie.js"></script>
<script src="bootstrap/Flot/jquery.flot.time.js"></script>
<script src="bootstrap/Flot/jquery.flot.stack.js"></script>
<script src="bootstrap/Flot/jquery.flot.resize.js"></script>
<!-- Flot plugins -->
<script src="bootstrap/js/flot/jquery.flot.orderBars.js"></script>
<script src="bootstrap/js/flot/date.js"></script>
<script src="bootstrap/js/flot/jquery.flot.spline.js"></script>
<script src="bootstrap/js/flot/curvedLines.js"></script>
<!-- jVectorMap
<script src="bootstrap/js/maps/jquery-jvectormap-2.0.3.min.js"></script>
bootstrap-daterangepicker
<script src="bootstrap/js/moment/moment.min.js"></script>
<script src="bootstrap/js/datepicker/daterangepicker.js"></script> -->
```

Figure 41 : Récupération des librairies dans une page html.

# 6. Description de l'application

## 6.1. Page d'accueil

La Figure 42 présente la page d'accueil de l'application web. Elle comporte une petite description de chaque outil de la plateforme et permet de passer vers la page d'inscription et la page d'authentification.

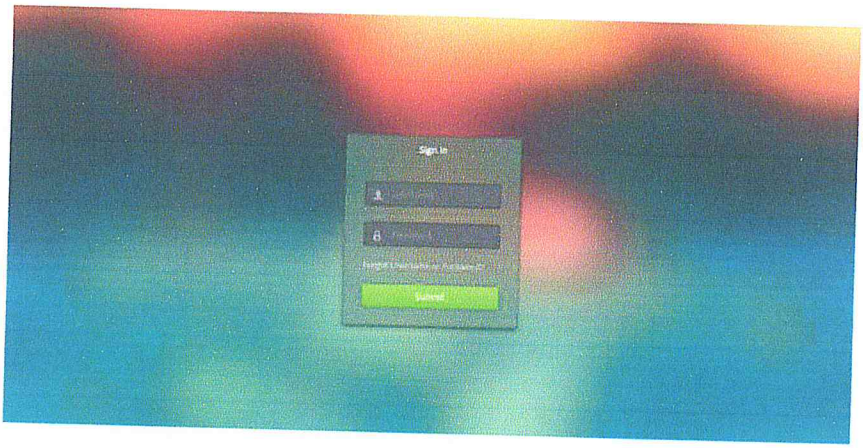

Figure 42 : Page d'accueil.

## 6.2. Page d'accueil « Utilisateur »

La page d'accueil d'un utilisateur est présentée sur la figure 43. Elle comporte un menu qui englobe les différentes options de contrôle et de gestion des utilisateurs, ainsi que l'historique des opérations.

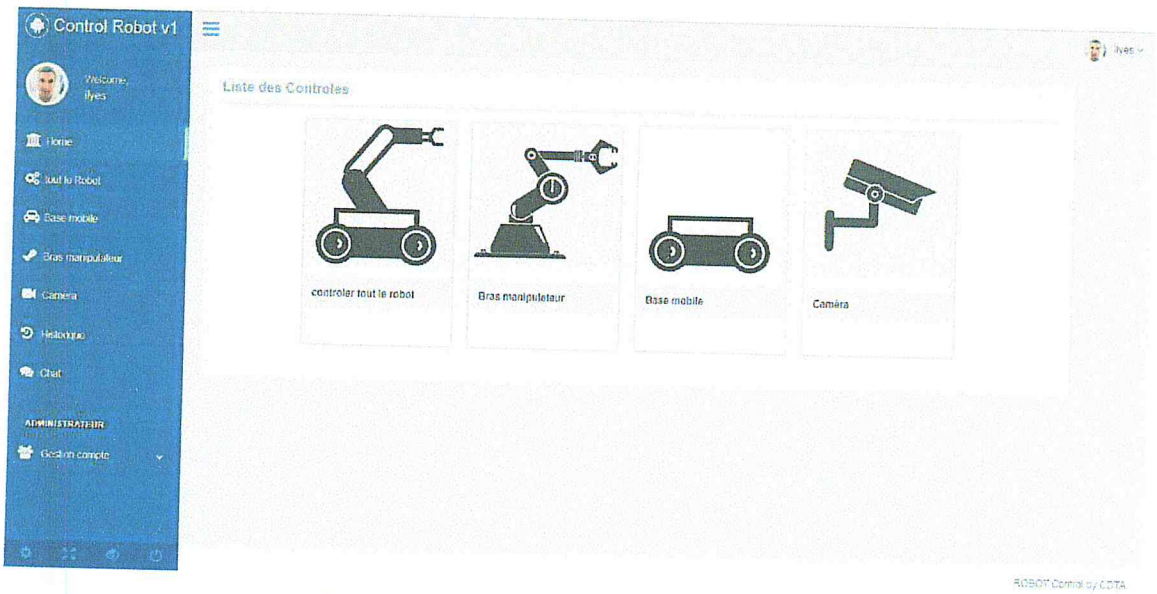


Figure 43 : Page d'accueil « Utilisateur ».

### 6.3. Page de contrôle « Tout le robot »

La page de contrôle de toutes les fonctionnalités du robot est illustrée sur la figure 44. Elle comporte une partie pour récupérer la vidéo de la caméra embarquée et les options de contrôle de vitesses et d'orientation de la base mobile. Enfin, elle offre la possibilité de contrôler les 6 axes du bras manipulateur, l'ouverture et fermeture de la pince.

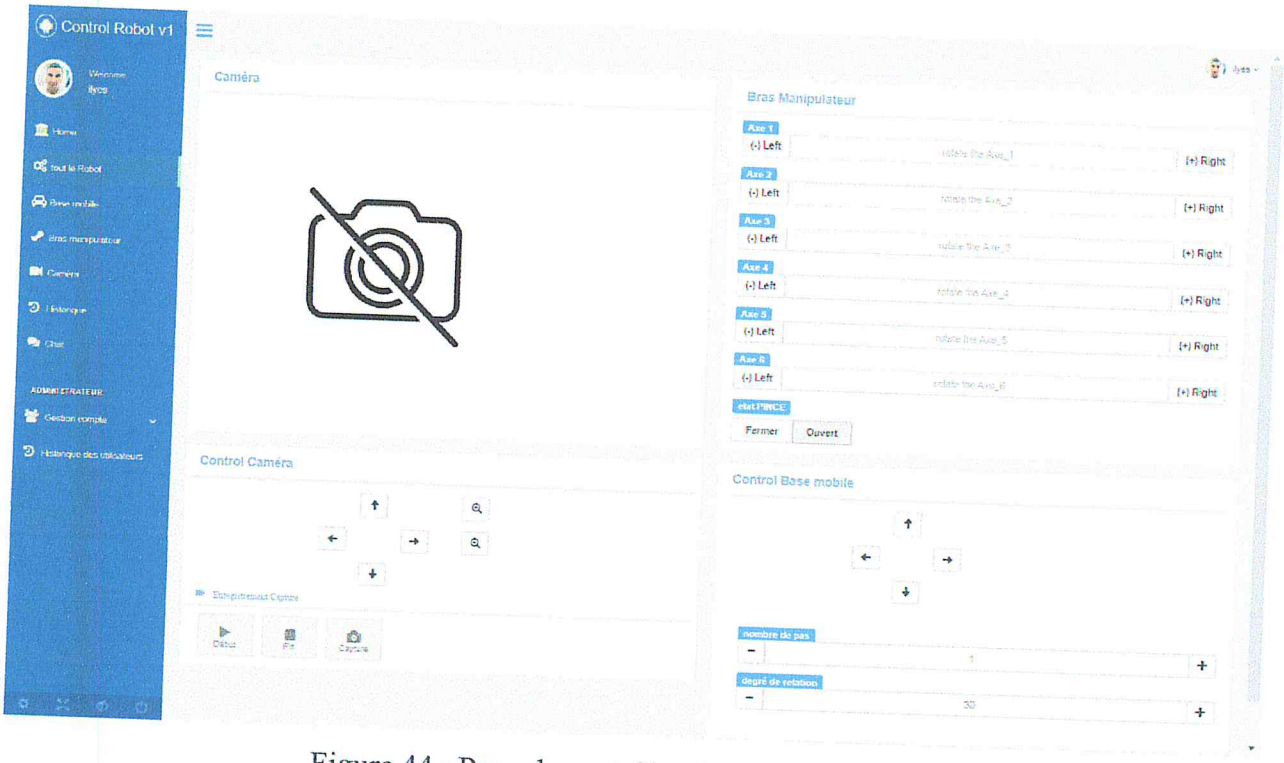


Figure 44 : Page de contrôle « Tout le robot ».

### 6.4. Page de contrôle de la «Base mobile »

La figure 45 représente la page de contrôle de la base mobile du robot. Elle comporte une partie pour récupérer la vidéo de la caméra embarquée et les options de contrôle des vitesses et de l'orientation de la base mobile. Enfin, la page permet d'afficher toutes les informations délivrées par les différents capteurs de la base mobile.

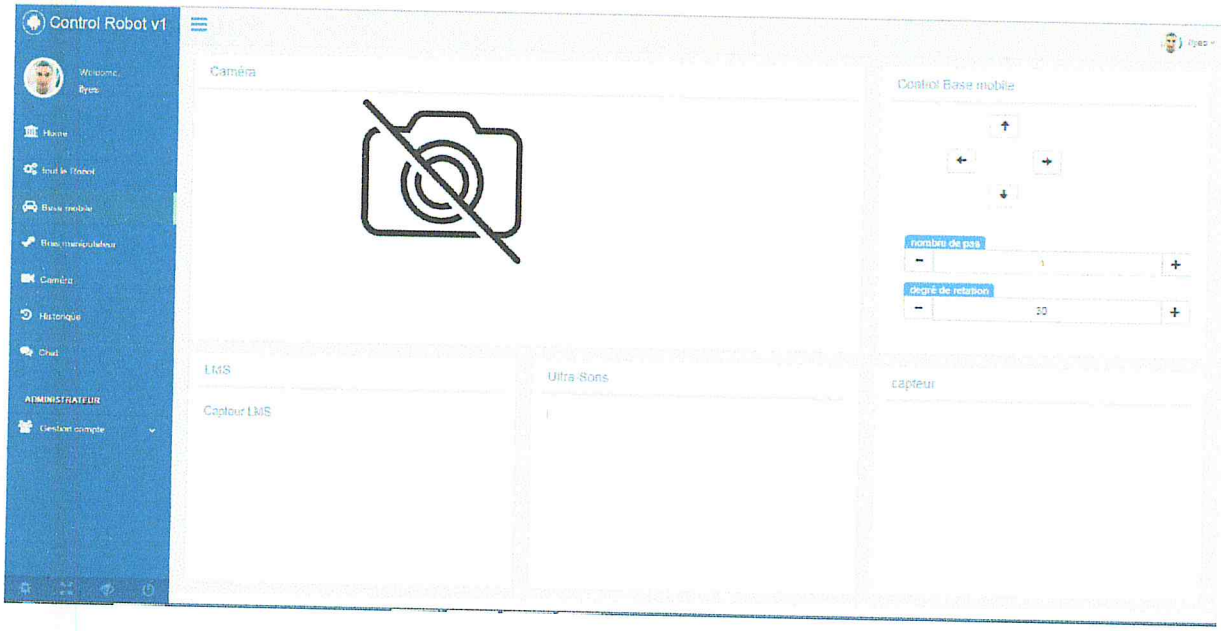


Figure 45 : Page de contrôle de la « Base mobile ».

## 6.5. Page de contrôle du « Bras manipulateur»

La page de contrôle du bras manipulateur du robot est représentée par la figure 46. Elle comporte ainsi une partie pour récupérer la vidéo de la caméra embarquée du robot. En outre, cette page comporte les options pour contrôler le mouvement de chacune des articulations du robot. Enfin, une dernière option permet le contrôle de l'ouverture et la fermeture de la pince.

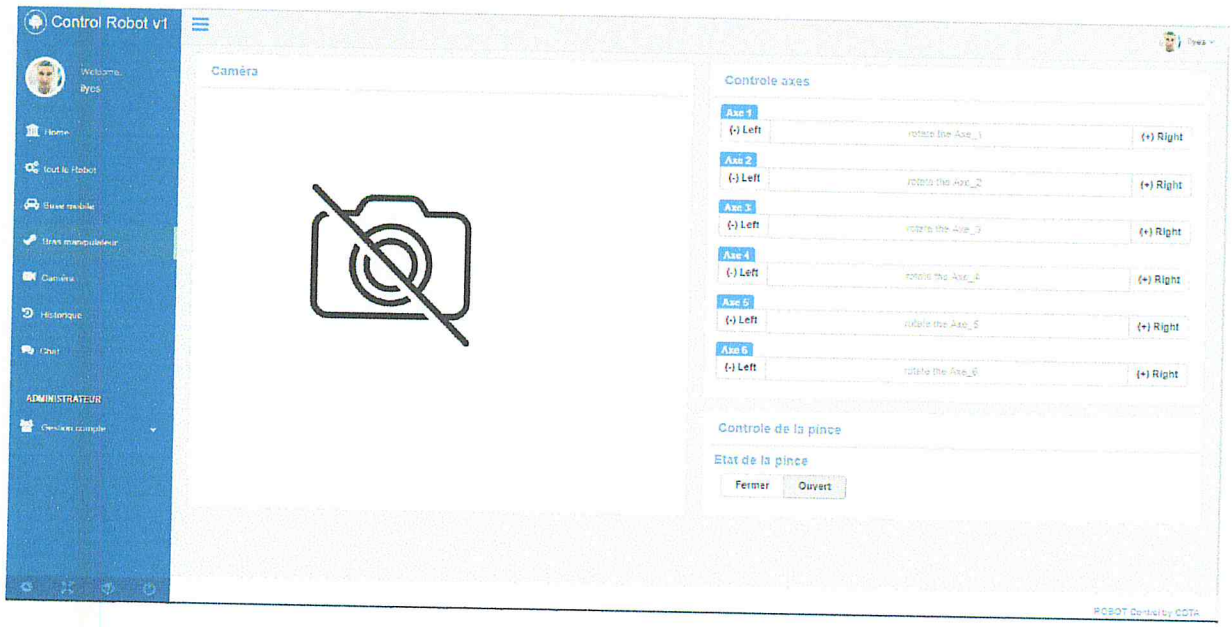


Figure 46 : Page de contrôle du « Bras manipulateur ».

### 6.6. Page de contrôle « Caméra»

La figure 47 représente la page de contrôle de la caméra embarquée du robot. Cette page comporte une partie pour récupérer la vidéo délivrée par la caméra de robot, des options pour changer son orientation, agrandir et dé-zoomer, enregistrer des vidéos, effectuer des captures d'images, etc.

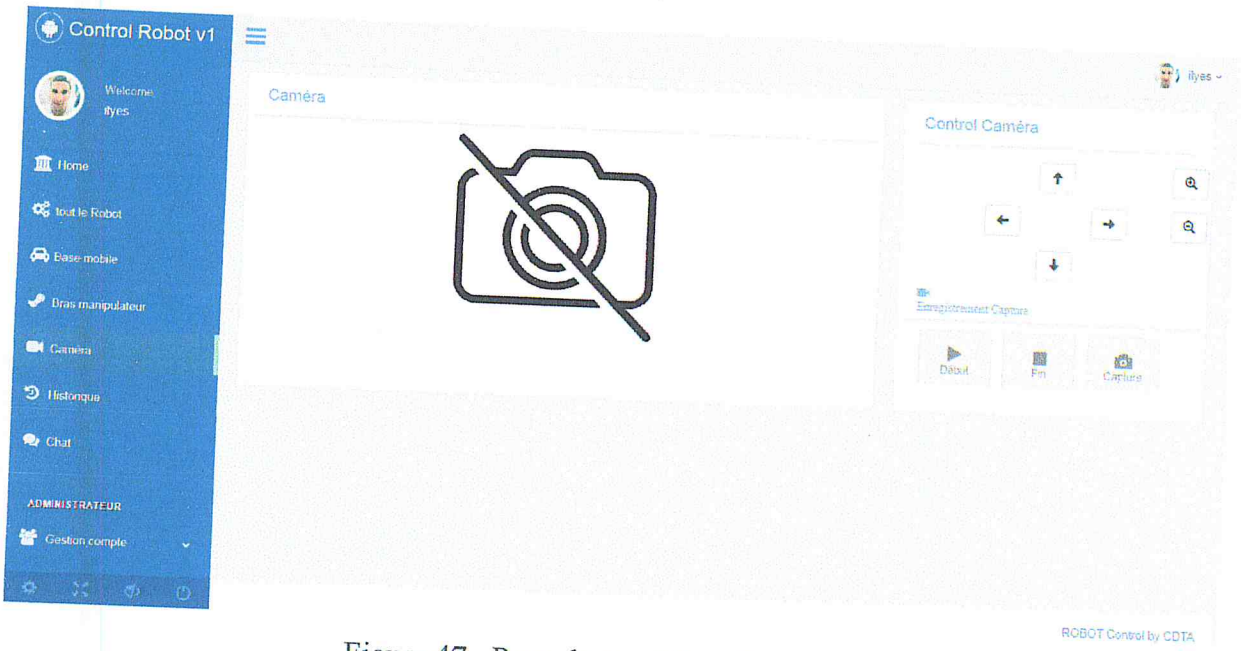


Figure 47 : Page de contrôle « Caméra ».

## 7. Tests expérimentaux et résultats obtenus

Nous avons réalisé une série de tests à travers l'envoi de quelques commandes. Les résultats obtenus ont permis de conclure que le WebRTC a démontré un gain en matière du temps de transmission et de réception des commandes.

Par exemple, l'envoi d'une simple commande de rotation d'un axe vers la gauche (Figure 48) :

- de l'application web vers l'application web côté passerelle via un canal WebRTC est inférieure à 1 milliseconde.
- de l'application web vers l'application java de la passerelle est de 3 millisecondes.

Ces résultats sont aussi obtenus grâce à la vitesse de transmission de Node.js. En observant ces résultats pour les différents tests effectués dans un réseau local, nous constatons une moyenne d'envoi de commandes d'environ 3 à 4 millisecondes. À cet effet, nous pouvons

confirmer que nous avons atteint nos objectifs de diminuer le temps de communication entre les différents pairs.

**Application web (coté user) :**

```
Sent Data: Rotate Axe_1_G H: 22 M: 46 S: 39 Mm: 452
```

**Application web (coté passerelle) :**

```
msg recieved : Rotate Axe_1_G H: 22 M: 46 S: 39 Mm: 452
```

**Lors de l'exécution :**

```
server_2 : Rotate Axe_1_G date : 10:46:39:454 PM CEST
```

Figure 48 : Résultats de tests obtenus.

Le tableau 1 ci-dessous montre les résultats obtenus lors de l'envoi d'une commande au robot dans le cas de notre application WebRTC par rapport aux anciens résultats obtenus dans [KH, 12]. Ces résultats montrent clairement que les temps nécessaires pour la communication entre le robot et l'opérant ont été considérablement diminués ce qui confirme que nous avons bien atteint les objectifs fixés par notre structure d'accueil à savoir le CDTA.

| | **Temps de connexion (millisecondes)** | | | | | |
|---|---|---|---|---|---|---|
| | **Connexion directe** | | **Connexion en local** | | **Connexion via internet** | |
| | **Base mobile** | **Bras manipulateur** | **Base mobile** | **Bras manipulateur** | **Base mobile** | **Bras manipulateur** |
| **[KH, 12]** | 8900 | 8200 | 10600 | 8900 | 12300 | 9400 |
| **WebRTC** | 3 | 3 | 3 | 3 | 3 | 3 |

Tableau 1: Comparaison des résultats des tests.

Dans le second cas, nous illustrons deux scénarios :

- Dans le premier, nous allons simuler un utilisateur qui contrôle à distance le robot pour exécuter une tâche de déplacement d'un objet d'un point à un autre. Dans ce cas, la durée totale d'exécution a été estimée à 10 secondes.
- Dans le deuxième cas, nous avons intégré plusieurs utilisateurs qui ont travaillé en collaboration pour effectuer la même tâche. Nous considérons que le premier

utilisateur contrôle le déplacement de la base mobile ; en même temps, le deuxième utilisateur configure le bras pour saisir l'objet. Par la suite, le premier utilisateur déplace le robot vers la position où le deuxième utilisateur doit déposer l'objet. Nous remarquons que la durée d'exécution de cette tâche partagée a été estimée à 4 secondes. Ainsi, nous pouvons conclure que le travail collaboratif entre plusieurs utilisateurs pour contrôlé un robot(MOSR) apporte un résultat meilleur par rapport à un seul en termes de temps d'exécution.

## 8. Conclusion

Dans un premier temps, nous avons présenté l'architecture de notre application web. Nous avons aussi décrit l'environnement de développement en présentant les langages de programmation et les technologies utilisés.

Par la suite, nous avons présenté globalement l'interface de l'application web de contrôle, via Internet, exploitant la technologie WebRTC. Grâce à cette interface, l'opérateur dispose d'un tableau de bord pour le contrôle à distance de tout ou d'une partie du robot.

Après, nous avons procédé à quelques tests de validation et simulé divers scenarios. Ainsi, la connexion au robot (base mobile et bras manipulateur) a été testée dans le cas d'une (*i*) connexion directe, (*ii*) connexion via le réseau local du CDTA, (*iii*) et connexion via le réseau Internet.

Enfin, dans le but de montrer la supériorité des résultats obtenus, nous les avons comparés avec ceux d'un travail similaire effectué dans [KH, 12].

# Conclusion générale

L'une des principales difficultés associées à la conception d'un système de contrôle télérobotique de robots est la latence et le délai de retard lors de l'exécution des commandes de manipulation de robots à distance. Un autre problème majeur est lié à la gestion de la file d'attente des opérateurs et des tâches dans le cas d'un contrôle du robot par plusieurs opérateurs simultanément.

Dans le cadre de ce projet, nous avons présenté une contribution au domaine de la télérobotique en traitant ces deux problèmes à travers la conception et le développement d'une application web pour le contrôle MOSR du robot RobuTER/ULM. Dans ce cas, nous nous sommes intéressés principalement à diminuer le temps d'exécution des tâches via la diminution de temps de connexion au robot en s'intéressant particulièrement aux solutions RTC (Real Time Communication) et plus précisément, à la technologie WebRTC et la plateforme logicielle Node.js pour l'exécution rapide des programmes JavaScript côté serveur. Dans le but d'offrir la possibilité de contrôler le robot par plusieurs opérateurs simultanément, nous avons développé un système de priorité qui gère les tâches simultanées et un système de chat entre les opérateurs.

Pour la conception de cette application web, nous avons utilisé le langage UML qui permet de prendre en compte les spécificités des applications web à travers ses différents diagrammes (classes, cas d'utilisation, séquence et déploiement).

Pour l'implémentation de notre application web, nous avons utilisé le framework Bootsrap pour sa création, et le langage JavaScript pour implémenter les APIs du WebRTC, ceux-ci du côté client. Pour le côté serveur, nous avons utilisé la plateforme logicielle node.js pour l'exécution rapide des programmes JavaScript et préserver ainsi une haute performance de cette application.

L'application développée offre aux opérateurs une interface web accessible via Internet pour le contrôle du robot. Une fois connecté, l'opérateur doit s'authentifier pour avoir accès à différentes commandes à exécuter par le robot. Ces commandes sont classées en trois catégories : contrôle du bras manipulateur, contrôle de la base mobile et contrôle de la caméra. De plus, l'interface permet à l'utilisateur de visualiser l'environnement du robot via l'affichage de la vidéo issue de la caméra embarquée du robot. Cette caméra peut être

commandée via cette interface offrant ainsi une souplesse de visualisation de l'environnement.

Pour ce qui est du fonctionnement de l'application web, elle offre un bon nombre de fonctionnalités intéressantes telles que :

- Contrôle en temps réel du robot (bras manipulateur, base mobile, caméra).
- Communication entre les utilisateurs (à travers des messages).
- Gestion des comptes utilisateurs.
- Consultation de l'historique des tâches exécutées.

Enfin, le contrôle télérobotique via le web ouvre plusieurs perspectives pour des applications intéressantes. L'application web développée constitue une plateforme de contrôle de robots mobiles, manipulateurs ou manipulateurs mobiles. Nous envisageons comme suite de ce travail l'exploitation de cette plateforme pour le développement de différentes versions en ajoutant :

- un système de priorités des utilisateurs lors d'un travail collaboratif sur le robot.
- un système de communication (chat) complet (appel audio et vidéo).
- la possibilité d'envoyer et de recevoir des messages (chat) hors ligne.
- une meilleure gestion du système de partage de flux vidéo (caméra) aux utilisateurs.
- les feedback des différents composants du robot (capteur LMS senor, Ultrasonic sensors).

# Références bibliographiques

[WRTC, 17] http://webrtc.org. (n.d.).

[ACTS, 14] Álvaro, R., Correia, A. M., Tan, F. .., & Stroetmann, K. .. (2014). New Perspectives in Information Systems and Technologies. *Springer Science & Business Media*.

[BZA, 16] Buffa, M., Zucker, C. F., Bergeron, T., & Aouzal, H. (Oct 2016). Semantic Web Technologies for improving remote visits of museums, using a mobile robot. Proceedings of the ISWC 2016 Posters & Demonstrations Track co-located with 15th International Semantic Web Conference (ISWC2016) ,Kobé,Japan.CEUR.

[BCFLH, 98] Burgard, W., Cremers, A., Fox, D., Lakemeyer, G., HÄHNEL, D., Schulz, D., et al. (Juillet1998). Le musée interactif robot de tour-guide.Dans :La15e conférence nationale sur léintelligence artificielle,Etats-Unis.

[CKOKMT, 2000] Chong, N., Kotoku, T., Ohba, K., Komoriya, K., Matsuhira, N., & Tanie, K. (Avril 2000). Distance contrôles coordonnés dans plusieurs coopération telerobot. Dans: La Conférence internationale sur la robotique et l'automatisation (ICRA 2000), San Francisco, Etats-Unis .

[GLM, 10] Germain, M., Liverneaux, P., & & Missana, M. C. (2010). Microchirurgie avec le robot Da Vinci S. La télémicrochirurgie: l'essor imminent. E-mémoires de l'Académie Nationale de Chirurgie, Paris, 9, 74-77.

[Irobot,03]https://www.army-technology.com/projects/irobot-510-packbot-multi-mission-robot. (n.d.).

[JB, 12] Johnston, A. B., & Burnett, D. C. (2012). WebRTC: APIs and RTCWEB protocols of the HTML5 real-time web. *Digital Codex LLC*.

[KTC, 17] K., C., T, A., & C, F. (2017). Usability Test in Different Types of Control-Authority Allocations for Multi-Operator Single-Robot System OCTOPUS. In: Ahram T., Falcão C. (eds) Advances in Usability and User Experience. AHFE 2017. Advances in Intelligent Systems an. *607*.

[Keyvan, 95] keyvan hashtrudi-zaad, M. u. (1995). *Ph.D thesis, design, implementation and evaluation of stable bilateral teleoperation control architectures for enhanced telepresence* .

[KH, 12] Khiter, B. H. ((2012, October)). Internet-based telerobotics of mobile manipulators: application on RobuTER/ULM. In International Conference on Intelligent Robotics and Applications . *((pp. 635-644)).*

[PMSGLS, 09] Pandya, S., Motkoski, J. W., Serrano-Almeida, C., Greer, A. D., Latour, I., & Sutherland, G. R. ( 2009, December 1). "Advancing neurosurgery with image-guided robotics".Journal of Neurosurgery. 111 (6): 1141–1149.doi:10.3171/2009.2.JNS081334. Retrieved 2011-05-04.

[RP, 08] Reed, K., & Peshkin, M. (2008). *Physical collaboration of human-human and human-robot teams. IEEE Transactions on Haptics 1(2), 108–120.*

[Roques, 08] Roques, P. (2008). ; UML 2: modéliser une application web, Les Cahiers du programmeur . Eyrolles.

[SG, 01] Sung, G. T., & Gill, I. S. (2001). Robotic laparoscopic surgery: a comparison of the da Vinci and Zeus systems, In Urology, Issue 6 , ISSN 0090-4295, https://doi.org/10.1016/S0090-4295(01)01423-6. *58*, 893-898.

[Con, 02] Jim Conallen. *Building Web applications with UML*(2002),. Addison-Wesley Longman Publishing Co., Inc., p 236-239.

[SAF,18]https://www.safaribooksonline.com/library/view/real-time-communication-with/9781449371869/ch01.html

[ALP] http://www.instructables.com/id/Aleph-10-Internet-Controlled-Microscope-Robotic-Ma/

[Diploi, 2000] https://laurent-audibert.developpez.com/Cours-UML/?page=diagrammes-composants-deploiement

[Sub, 13] https://www.sublimetext.com/blog/articles/sublime-text-3-point-0

[Netb, 16] https://netbeans.org/community/releases/82/install.html

[Cmdr, 16] http://black-unicorn.codeur.online/cmder-une-meilleur-console-pour-windows/

[Wamp, 18] http://www.wampserver.com/

[Bot, 12] https://agency-inside.com/2016/06/definition-webmarketing-bootstrap/

[ST, 18] https://www.lavachequicode.fr/introduction-au-webrtc