République Algérienne Démocratique et Populaire.
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique.

Université Saad Dahlab, Blida
USDB.

Faculté des sciences.
Département informatique.

Mémoire pour l'obtention
d'un diplôme d'ingénieur d'état en informatique.
Option : IA

Sujet :

CONTROLE D'UN ROBOT MANIPULATEUR MOBILE BASE SUR UNE ARCHITECTURE CLIENT/SERVEUR

Présenté par : LAIB LAHCENE
BESSAIH REDOUANE

Promoteur : KADRI Mohamed
Encadreur : BOUCHEMMA Rafik

Thème proposé par :
Equipe Système Robotisé de Production
Division Robotique et Productique (CDTA)

Soutenue le: 02/10/2005, devant le jury composé de :

M. BALA	**Université Saad Dahlab, Blida USDB**	**Président**
M. MAHIEDDINE	**Université Saad Dahlab, Blida USDB**	**Examinateur**

PROMOTION 2004/2005

Dédicace de hcene

Je dédie premièrement ce mémoire à mes parents qui m'ont soutenu tout au long de ma vie, qui m'ont guidé et orienté dans le bonne voie .je les remercie pour m'avoir aidé à surmonter les obstacles.

Et aussi mes frères, et a mes cher amis Mazouzi Abdelghani,Rabie ,Samir ,Nabi, Hafid, Amine,l ainsi que tous les autres

Dédicace de Redouane

Je dédie premièrement ce mémoire à mes cher parent qui m'ont soutenu depuis toujours et dans toutes les épreuves.

Je le dédie aussi à mes frères et soeurs, a ma tante et a ma belle soeurs, ainsi que toutes ma famille de prés ou de loin.

Enfin je le dédie à mes amis Marzak, Boussaad, Bilal, Rouji et tous les autres.

SOMMAIRE

CHAPITRE I

INTRODUCTION GENERALE

INTRODUCTION GENERALE

Le domaine de la robotique a connue une croissance énorme lors des dernières années. Les avancées scientifiques, le perfectionnement des systèmes mécaniques et l'augmentation de l'intégration ont permis de passer de la génération des robots manipulateurs fixes à celle des robots mobiles, puis à celle des robots manipulateurs mobiles. Pour une meilleure exploitation de ces systèmes, pendant longtemps les efforts ont été dirigés vers leurs maîtrise du point de vue de la régulation qui permet au robot de suivre au plus exact un comportement (des trajectoires) strictement connu à l'avance. Les difficultés résident, d'une part, dans l'écriture d'un modèle réaliste, c'est-à-dire exact, du robot, d'autre part, dans la découverte du type de contrôle adéquat, sachant qu'on doit faire face à un système très non linéaire.

L'apport de ce développement technologique a permis à la robotique de s'élargir à de nombreux domaines comme la médecine, le militaire, l'exploration. Du fait de cet élargissement les robots sont amenés à intervenir dans des milieux très différents tels que le milieu sous-marin, aérien, spatial (robots sondes d'exploration) ou encore le milieu terrestre.

Même si Jusqu'à maintenant, les coûts élevés ont limité leur utilisation dans l'aérospatiale, dans les domaines militaires, et dans les centrales nucléaires. Néanmoins, la prolifération des développements dans ce domaine permet l'introduction de cette technologie dans les domaines commerciaux de l'agriculture, et aussi dans l'industrie, des services, des mines, de la médecine et d'autres encore. C'est un secteur stratégique avec un fort potentiel de croissance.

Le fait que ces robots interviennent dans des milieux aussi variés et de plus en plus dans des endroits inaccessibles à l'homme a soulevé de nombreux problèmes liés à la commande de ces robots. L'éloignement de l'opérateur humain a impliqué de développer des systèmes de perception et de communication de plus en plus évolués, de doter les robots de facultés de décision et d'analyse permettant de suppléer partiellement à la décision fournie par l'opérateur. Tout cela a abouti à la création de robots commandés à distance avec différents niveaux d'autonomie.

La commande d'une machine, d'un outil à distance rend souvent la vie de l'opérateur plus simple, plus pratique. L'homme à par exemple, depuis longtemps voulu commander un véhicule sans devoirs se trouver à l'intérieur. Dans l'absolu, la commande à distance intervient lorsque l'opérateur ne peut agir lui-même, pour des raisons de sécurité (endroits exposés à de la radioactivité), d'inaccessibilité (tuyaux d'aération), d'agressivité du milieu (espace, grandes profondeurs),...

La recherche en robotique s'est quelque peu réorientée par suite des échecs sur l'autonomie qui ont obligé à de nouvelles réflexions. De la nous pouvons dire qu'opter pour l'autonomie des robots serais très difficile a concevoir sans garantie de résultat, pour cela beaucoup ont opté pour le contrôle a distance du robot.

Les systèmes commandés par des réseaux informatiques sont un domaine en pleine effervescence au sein de la communauté scientifique vu l'apport de la nouvelle technologie (réseaux informatique, Internet) et les avancées en termes des performances des ordinateurs (processeurs, mémoires.....) rend plus facile le contrôle à distance des robots et l'exécution des tâches en temps réels. On peut ainsi contrôler des robots (inspection, déminage, sauvetage, médecine.) ou même des équipements lourd comme le télescope, le contrôle distant étant rendu possible par l'introduction d'un réseau dans la boucle de commande du système cybernétique. Tout cela induit des problèmes spécifiques tels:

-Ergonomie du poste de travail (élaboration de la consigne).

-Qualité des retours informationnels (réalité vertuelle et augmentée).

-Retard de transmission des informations (dus au réseau).

-Sécurité de l'application.

Face à la diversité, l'hétérogénéité et la complexité des systèmes à mettre en oeuvre, il est nécessaire de doter le robot d'une architecture de commande permettant de coordonner et de piloter l'ensemble. Les architectures de commande remplissent plusieurs fonctions. Elles permettent tout d'abord de conférer au robot un comportement spécifique permettant l'accomplissement de sa mission. Pour cela, elle met en oeuvre tout ou partie des équipements composant le robot en organisant et gérant les ressources de ce dernier. D'autre part, l'architecture octroie au robot une part d'autonomie fonctionnelle et/ou décisionnelle permettant à ce dernier d'interagir avec son environnement. Enfin, l'architecture de commande doit permettre de superviser les actions entreprises par le robot par l'opérateur avec les contraintes générées par l'environnement.

Parmi les travaux entrepris dans cet axe de recherche, nous pouvons citer les travaux de Arnaud Leleve [1] de l'université de Montpellier sur le télépilotage à longue distance d'engins mobiles robotises pour des tâches d'intervention en milieu difficile ou hostile pour l'homme (nucléaire, spatial, sous-marins,...). Ses travaux ont permis de mettre en place un système de communication entre un poste de télépilotage Assisté par Ordinateur et un engin mobile robotisé (manipulateur mobile constitué d'une base Andruet et d'un bras manipulateur de type Puma).

Vicente Egea and all [2] ont pu concevoir un véhicule autoguidé en utilisant sur une commande basée sur une architecture client/serveur pour permettre un contrôle à distance de leur véhicule.

Nous souhaitons réaliser une architecture de commande à distance pour un robot manipulateur mobile (Robuter-ULM du CDTA). Nous partons du postulat que le robot possède à l'intérieur de sa chaîne de contrôle un opérateur humain. La commande repose sur une architecture client/serveur, le serveur tourne sur le Pc embarqué du robot manipulateur mobile, et le client un Pc hôte doté d'une interface

utilisateur qui permet à l'opérateur de percevoir l'état du robot et d'envoyer ces consignes vers le serveur pour qu'ils soient exécuter par le robot.

Nous allons donc présenter dans le deuxième chapitre le site expérimental sur lequel nous comptant implémenter notre architecture de commande. Nous présenterons dans les détails les caractéristiques du robot manipulateur mobile (Robuter-ULM du CDTA)

Dans le troisième chapitre, le principe de la commande à distance sera exposé en plus des différents outils informatiques se reportant à l'architecture réseau utilisée.

Dans le quatrième chapitre, nous présenterons la méthodologie de conception de notre application en se basant sur le langage de modélisation UML.

Enfin, dans le dernier chapitre, nous mettrons en œuvre notre architecture de commande sur robot Robuter-ULM. L'application réalisée fera l'objet d'une présentation en illustrant ses diverses fonctionnalités.

Notre document se verra terminer par une conclusion générale.

CHAPITRE II

PRESENTATION DU ROBOT ROBUTER-ULM

I. Introduction

Le Robuter-ULM [3] montré en figure 1 est un robot manipulateur mobile constitué d'une base mobile à 4 roues et d'un bras manipulateur à six axes avec une pince électrique à son extrémité. Le robot est équipé d'une ceinture de capteurs ultrasons, d'une Camera CCD et d'un capteur d'effort. Le système est commandé par un Pc industriel embarqué MMX 233 et quatre cartes Robosoft à base de microcontrôleur MPC555.

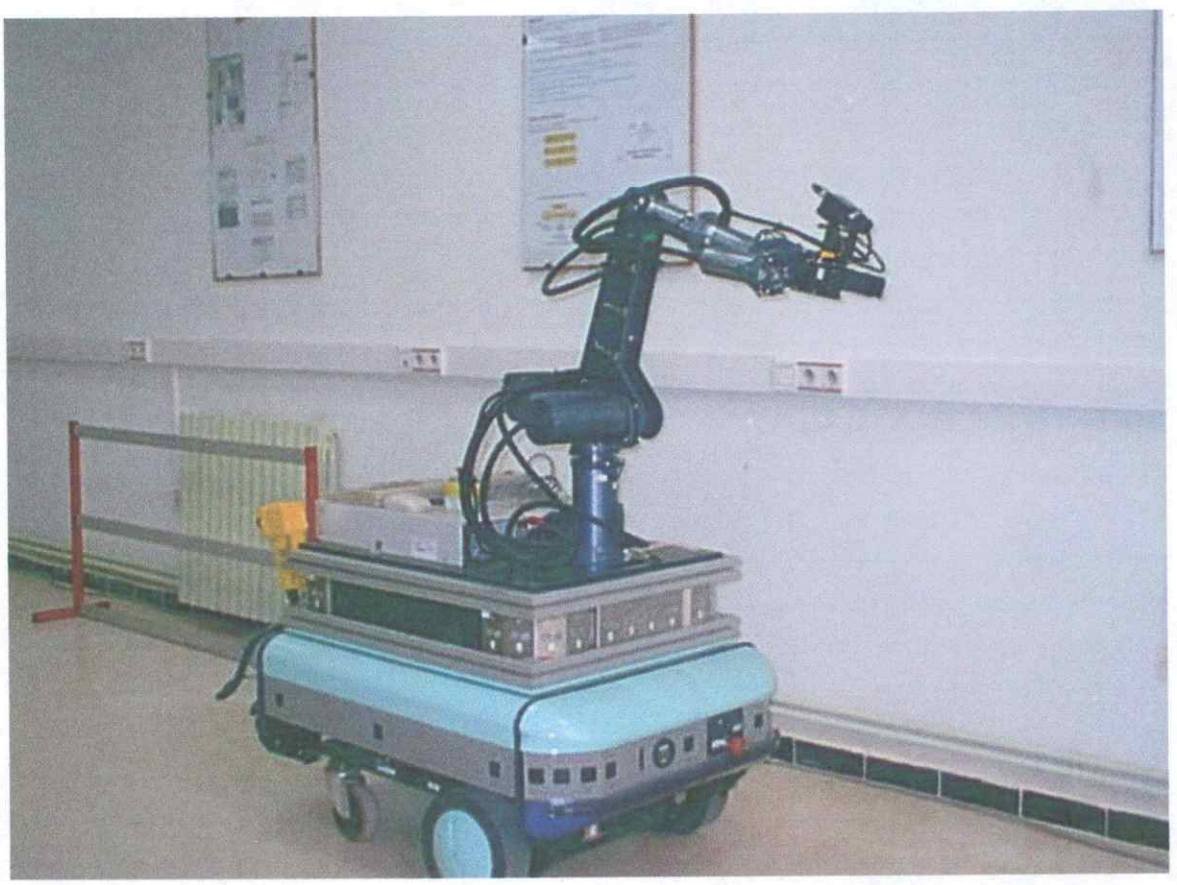


Figure 1 : Robot Robuter-ULM du CDTA

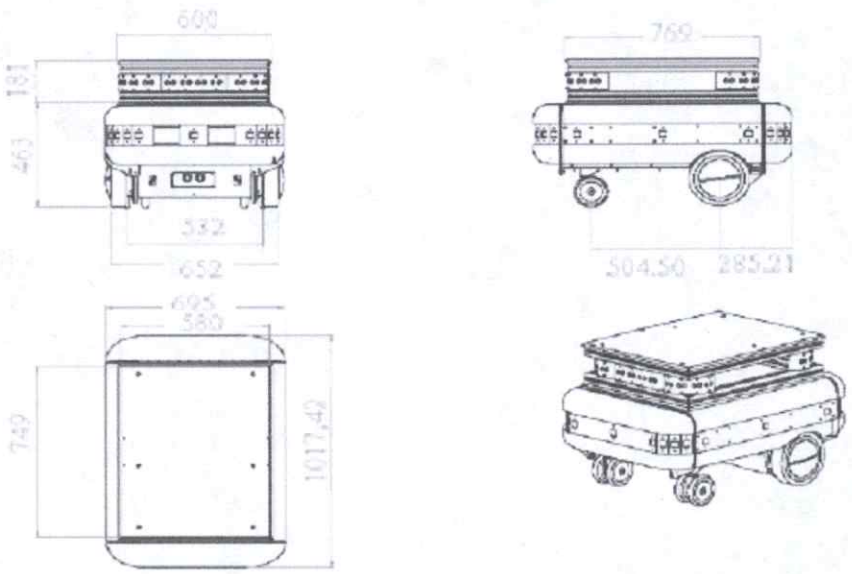
II. Description du Router-ULM

II.1. La base mobile (Robuter)

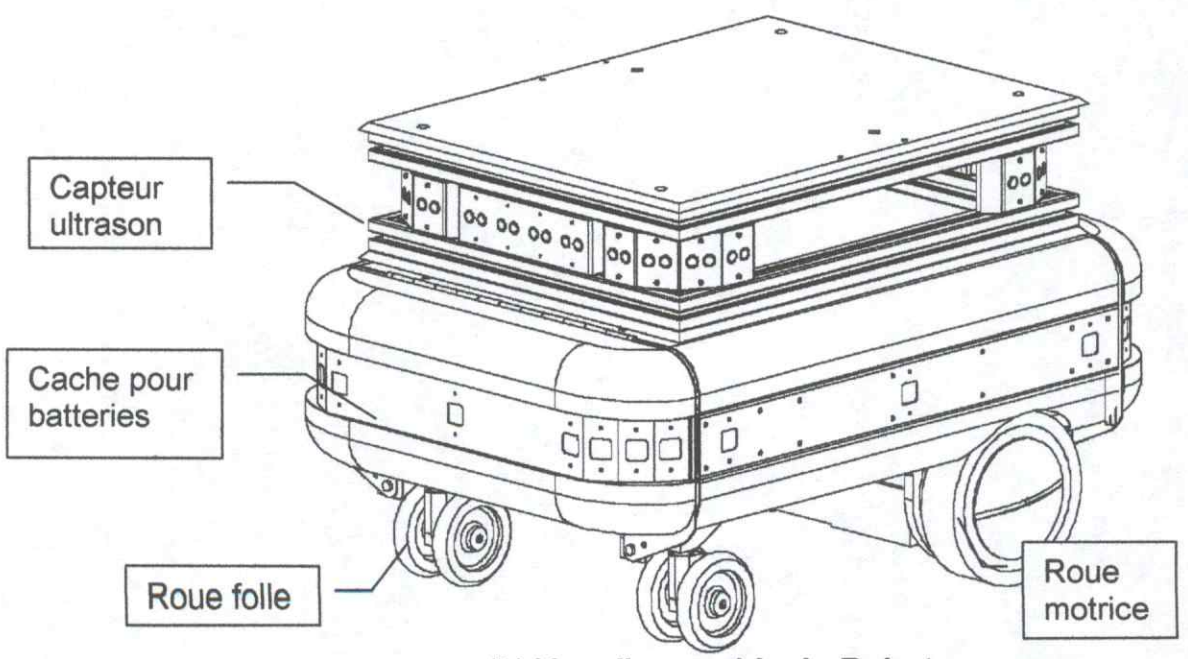
La base mobile comme le montre la Figure 2 est une plate forme à 4 roues, de dimension 695 mm par 1017 mm. Elle pèse environ 150 kg, d'une capacité de charge de 120kg. Deux des 4 roues sont motrices d'une capacité de charge de 15 kg, elles sont actionnées par deux moteurs électriques de 2*300 W à courant continue, sa vitesse varie entre de 5cm/s à 1.25 m/s, avec des roues de 250 mm de diamètre, et un couple nominal de 22 Nm par roue. Les deux autres roues sont des roues folles qui assurent la stabilité de l'ensemble de la plate-forme. La base est équipée de 4

batteries de 12 volts chacune qui fournissent l'énergie nécessaire au fonctionnement du robot.

Le Robuter est une base mobile de type différentielle, cela veut dire que l'orientation de la plate-forme se fait par différences de vitesse des deus roues motrices. La précision de position par encodeurs optiques, et un système d'odomètrie donne la position relative du robot par rapport à un repère lié à la plate-forme.



(a) Vue du Robuter



(b) Vue d'ensemble du Robuter

Figure 2 : Base Mobile (Robuter)

II.2. Les capteurs ultrasons

Le Robuter (base mobile) dispose d'une ceinture de 24 capteurs ultrasons du type SRF04 – Ultrasonic Range Finder, numérotés de 1 à 24. Le premier étant le plus a gauche de la ceinture avant lorsqu'on se met dans le sens du robuter. Ces capteurs servent à détecter les obstacles par envoi et réception d'ondes ultrasonores. Le capteur ultrason SRF04 comme le montre la figure 3 est très compacte de dimensions 43mm x 20mm x 17mm de hauteur. Il arrive à donner des mesures dans un intervalle de 3 cm jusqu'à 3m avec une ouverture angulaire de 30 degrés et une capacité de détecter un objet de 3 cm de diamètre à plus de 2 mètres de distance.

Les 24 capteurs ultrasons sont regroupés en trois groupes par un nombre de huit chacun. Chaque groupe est connecté à une interface digitale d'entrées/sorties à base de microcontrôleurs. Ces cartes sont directement reliées au Pc embarqué via une liaison RS232.

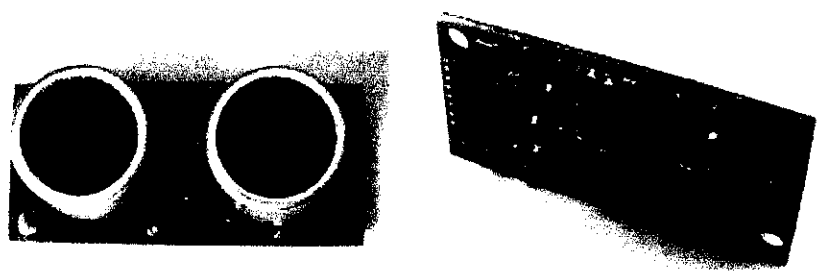


Figure 3 : Capteur Ultrasons

Les 24 capteurs sont disposés sur la plate-forme afin de permettre une totale couverture du robot vis-à-vis de son environnement. Le tableau ci-dessous fourni la position de chaque capteur par rapport au repère lié au robot mobile où X est un axe vers l'avant et qui se trouve dans l'axes du robuter, Y est un axe vers la droite et il se trouve dans l'axe des roues motrices. Par contre, Z est un axe vers le haut du véhicule et Thêta donne l'angle que fait l'axe perpendiculaire à chaque capteur avec l'axe X.

Tableau 1 : Positionnement Des Capteurs Ultrason.

Capteur	X	Z	Y	Theta
1	485.96	428.50	278.50-	90
2	535.98	428.50	-265.10	60
3	572.60	428.50	-228.48	30
4	586.00	428.50	-178.46	0

5	586.00	428.50	-90.00	0
6	586.00	428.50	-30.00	0
7	586.00	428.50	30.00	0
8	586.00	428.50	60.00	0
9	586.00	428.50	178.46	0
10	572.60	428.50	228.48	330
11	535.98	428.50	265.10	300
12	485.96	428.50	278.50	270
13	-39.96	428.50	278.50	270
14	-89.98	428.50	265.10	240
15	-126.60	428.50	228.48	210
16	-140.00	428.50	178.46	180
17	-140.00	428.50	90.00	180
18	-140.00	428.50	30.00	180
19	-140.00	428.50	-30.00	180
20	-140.00	428.50	-90.00	180
21	-140.00	428.50	-178.46	180
22	-126.60	428.50	-228.48	150
23	-89.98	428.50	-265.10	120
24	-39.96	428.50	-278.50	90

II.3. Le bras manipulateur

Le bras manipulateur (ULM) (figure 4) est un bras ultras léger avec 6 axes, muni d'une pince électrique à deux doits à son extrémité. Elle fonctionne en tout ou

rien avec un signal de commande et deux signaux d'état de la pince. Le bras est constitué de segments, de dimensions données dans la figure 5, reliés entre eux par des articulations. La porte du bras est de700 mm avec une répétitivité de +/- 1 mm .Sa capacité de charge est de 2 kg lorsque que le bras est totalement déployé. Le bras est constitue de 7 moteurs DC ,6 pour les 6 axes et un pour la pince. Le mouvement de chaque axe est détecté par un capteur de position (encodeur). Sur chaque axe on trouve une butée électrique et une autre mécanique qui apporte une sécurité au bras contre une mauvaise utilisation au-delà du débattement angulaire permis donnés dans le tableau ci-dessous.

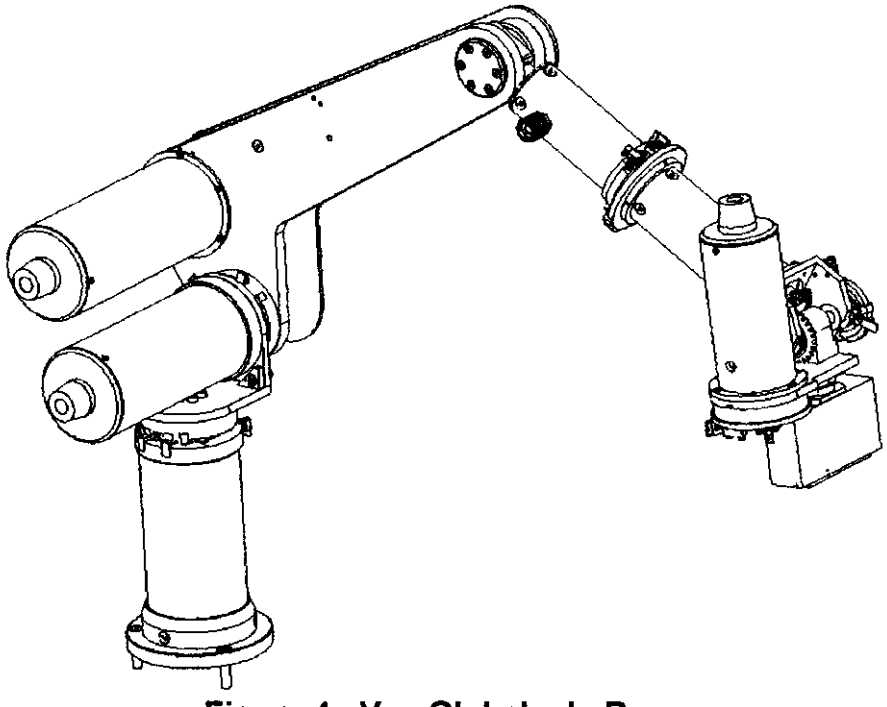


Figure 4 : Vue Globale du Bras

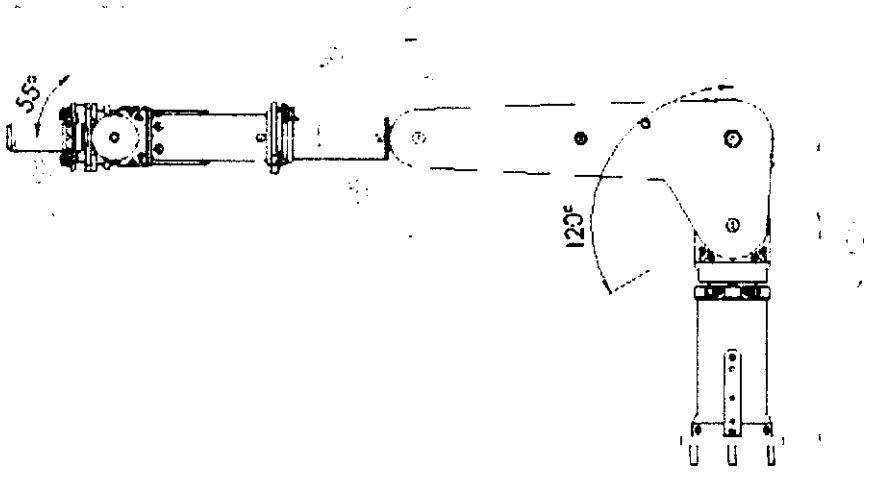


Figure 5: Dimensions du Bras

Tableau 2 : Débattement des Axes

AXE	angle min	angle max
Axe1	-95.4	95.4
Axe2	-22.8	81
Axe3	+10	168
Axe4	-75.5	78
Axe5	-55	55.5
Axe6	-96.8	96.8

II.4. Capteur d'effort

Un capteur d'effort est placé sur l'organe terminal du robot manipulateur avant la pince. Le capteur d'effort a pour fonction de fournir les forces et les couples exercés par le robot sur son environnement. L'importance d'un tel dispositif est de permettre l'implémentation d'une commande en force. Ce genre de commande est fortement présent dans certaines applications en robotique qui demande une interaction entre le robot et son environnement.

Le capteur placé sur le bras UL est de type Gamma SI-32-2.5 de ATI Industrial Automation. C'est un capteur 6 axes qui permet la mesure les trois forces selon les axes X, Y et Z ainsi que les trois couple autour de ces trois axes. Le fonctionnement de ce capteur est basé sur le principe piézoélectrique qui permet de transformer à l'aide d'un transducteur toute force ou couple en un signal électrique.

II.5. Caméra

Une caméra de type XC-ST50CE (figure 6) de chez Sony placée sur l'organe terminal du bras. Cette caméra est compacte avec une taille de 44 (l) x 29 (h) x 57,5 (p) mm et légères (seulement 110 g). C'est caméra noir et blanc offrent des images d'excellente qualité et une très grande sensibilité. Elle est dotée de la dernière génération de capteurs d'images CCD Sony. Elle est idéale pour les applications exigeantes de l'industrie, de la microscopie, du traitement d'images et de la vision industrielle.

Figure 6 : Caméra XC-ST50CE

La caméra est connectée à une carte d'acquisition de type PCI Frame Grabber Modèle 611. La carte d'acquisition placée sur le bus PCI. Elle permet la capture des images monochrome qui proviennent de la caméra et stocker dans la mémoire RAM du Pc embarqué.

II.6. LAN sans fil

Afin de garantir une liberté de déplacement, une liaison sans fil entre le robot et le point d'accès au réseau est nécessaire. Pour réaliser cette liaison on a utilisé un produit de l'entreprise BREEZECOM : le BreezeNET PRO.11. Il se compose de deux produits qui réalisent une transmission transparente point à point entre le point d'accès au réseau (par le BreezeNET PRO.11 AP-10) et le robot (par le BreezeNET PRO.11 SA-10). La configuration est visible sur la figure 7.

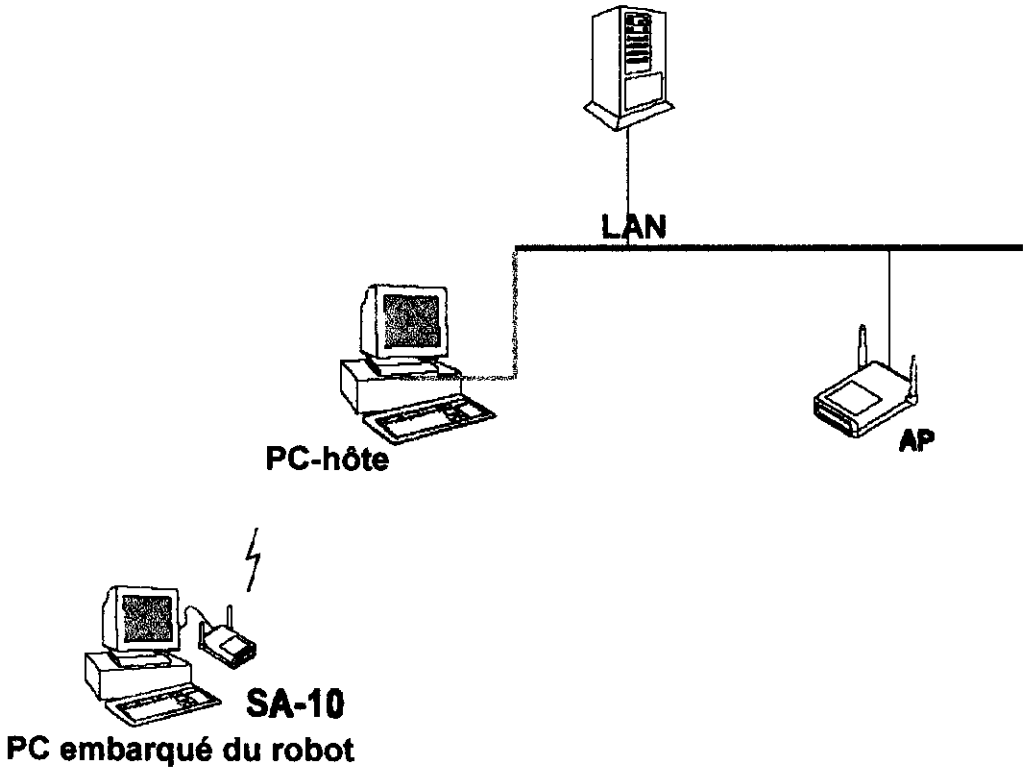


Figure 7: Système LAN sans Fil

III. Architecture matérielle du robot

Le robot manipulateur mobile possède une architecture matérielle (figure 8) distribuée comprenant quatre cartes Robosoft à base de microcontrôleur MPC555 de Motorola et d'un Pc embarqué. Les deux moteurs de la base mobile sont contrôlés par une carte MPC555, deux autres cartes permettent le contrôle des six axes du bras UL et de la pince. La dernière carte permet l'acquisition des mesures données par le capteur d'effort. Tout système est supervisé par un Pc embarqué de type industriel Intel MMX 233 qui permet à l'utilisateur l'accès aux différentes ressources du robot.

Les quatre cartes ainsi que le Pc embarqué sont reliés par un bus CAN qui permet l'échange du flux de données entres les différents processeurs. Mise à part les

capteurs ultrasons et la caméra toutes les données des ressources du robot transitent sur le bus CAN. Les capteurs ultrasons sont connectés directement au Pc embarqué via une liaison RS232. Quant à la caméra, un câble coaxiale la relié à la carte d'acquisition.

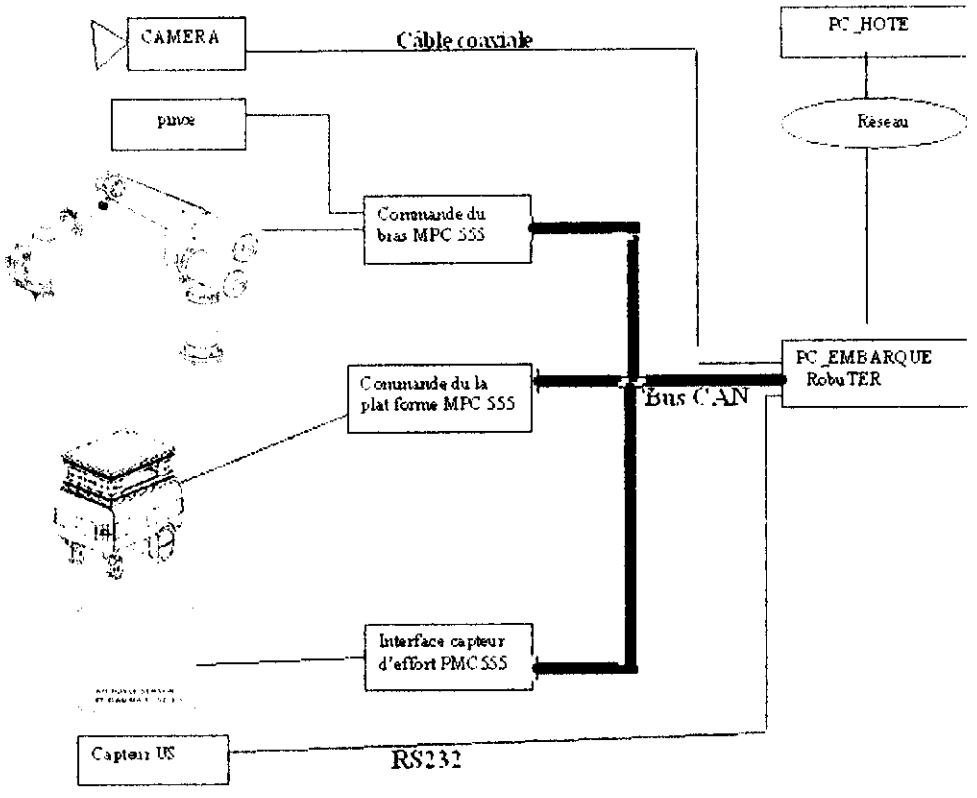


Figure 8 : Architecture matérielle du Robuter-ULM

III.1. Carte Robosoft MPC555

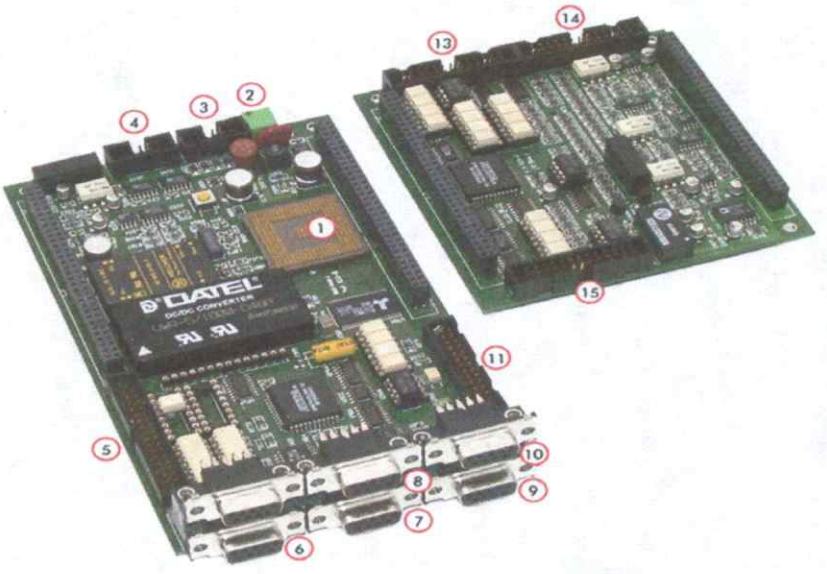


Figure 9 : Carte Robosoft MP C555

La carte Robosoft MPC555 est conçue pour la commande de robot, de véhicule électrique ou toute autre machine. Elle permet de commander un axe grâce à l'ajout d'une carte fille le contrôle s'étend jusqu'à 4 axes. Cette carte est basée sur le microcontrôleur MPC555 32-bit de Motorola. Elle peut opérer à une fréquence de 40 MHz. Divers signaux logique et analogique peuvent être utilisé pour l'acquisition de et la commande de matériel. Les sorties disponibles peuvent être analogique ou PWM. On y trouve :

(1) Connecteur du chip MPC555
(2) Alimentation continue
(3) Interface BDM (Basic Debug Interface)
(4) Entré Analogique (Joystick,…)
(5) Entré/Sortie logiques
(6) Ligne série Synchrone (Codeur absolu)
(7) Ligne série Asynchrone (Port 0)
8) Ligne série Asynchrone (Port 1)
(9) Connecteur du bus CAN (Port 0)
(10) Connecteur du bus CAN (Port 1)
(11) Axe 1
(13) Axe 2
(14) Axe 3
(15) Axe 4

IV. Architecture logiciel [4]

Afin d'exploiter les ressources du robot, l'application utilisateur doit contenir deux parties : une partie haut niveau en C/C++ et une autre bas niveau sous syndex avec

un échange de données entre les deux partie en utilisant la couche RTAI de linux comme le montre la figure 10.

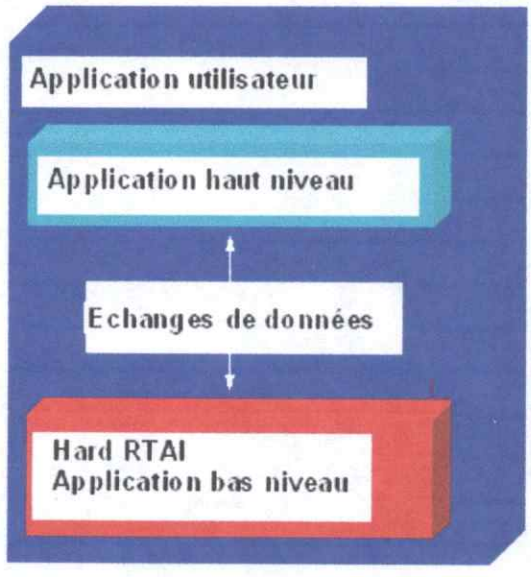


Figure 10 : Architecture Logicielle

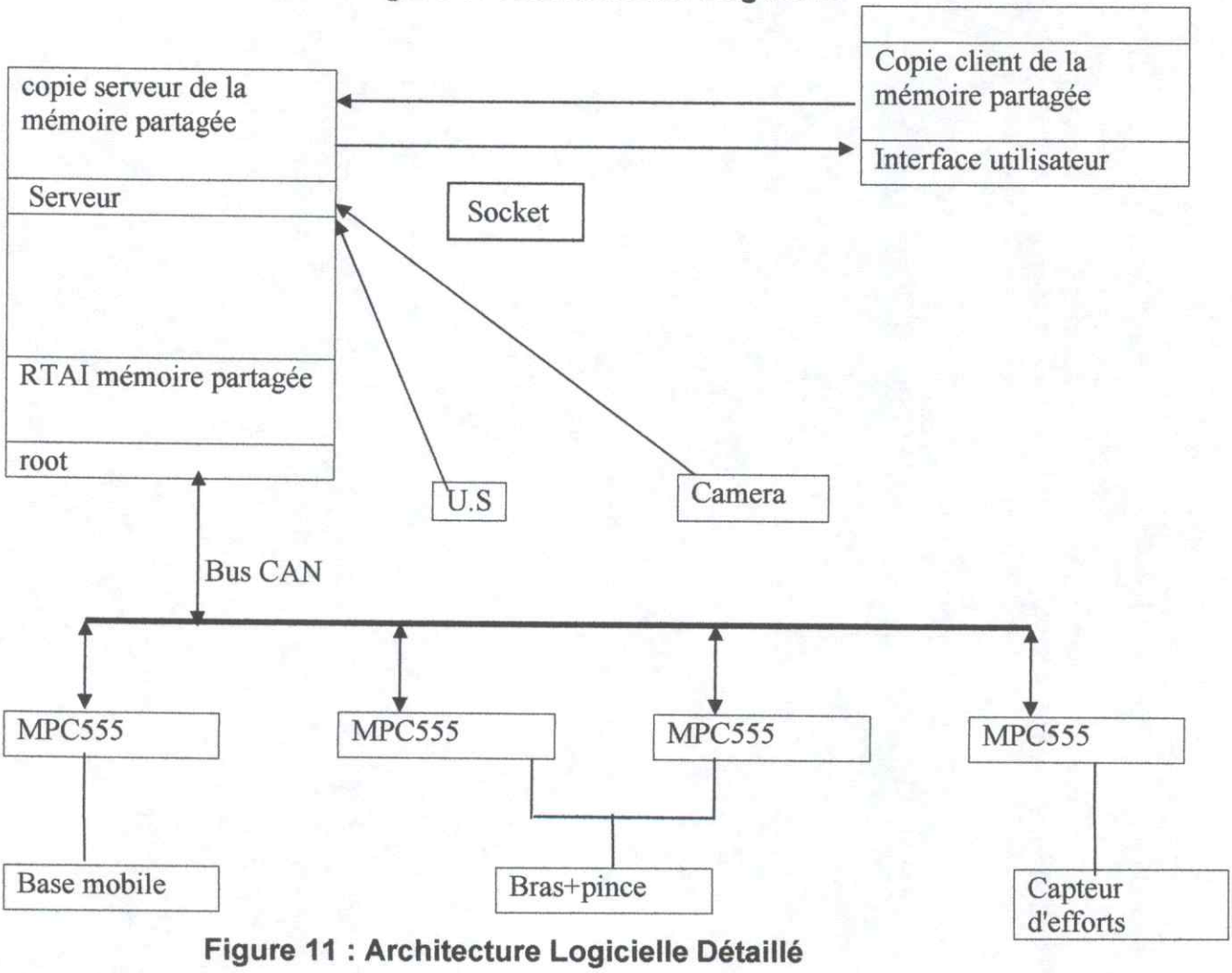


Figure 11 : Architecture Logicielle Détaillé

On entend par une application haut niveau tout programme comme la planification de trajectoires, stratégies de navigation, etc. …
L'échange de données se fait par le biais des mémoires partagées qui font parti du noyau RTAI et de l'espace utilisateurs de Linux.
Toute ressource présente sur la carte Robosoft peut être partagée sans aune restriction.

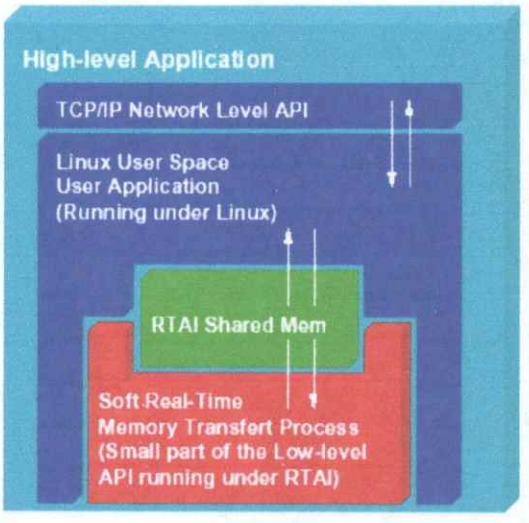


Figure 12 : Application Haut Niveau

Une application bas niveau est un programme conçu sous syndex comme la commande des moteurs, la lecture des capteurs, etc. …
Les ressources matérielles sont contrôlées par plusieurs cartes MPC555. Ces cartes s'échangent des données via le bus CAN. Par conséquent les entrées/sorties sont prises en charge par la source logicielle dédiée qui s'exécute dans les cartes MPC555 puis la couche RTAI est utilisée pour la mise à jour de mémoires partagées pour l'espace utilisateurs linux d'échange de données.

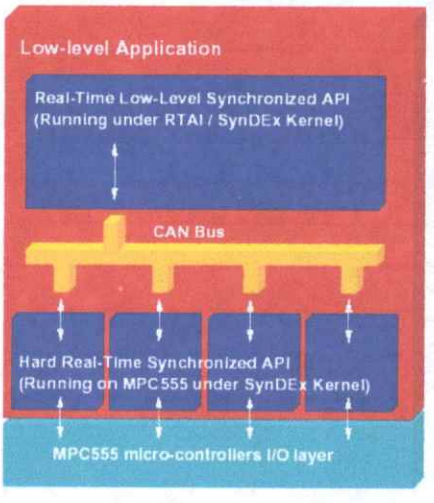


Figure 13 : Application Bas Niveau

IV.1 Méthodologie AAA

C'est une méthodologie fonde sur des modèles de graphes, autant pour spécifier les algorithmes applicatifs et les architectures matérielles destribuées, que pour déduire des implantations possibles en termes de transformation de graphes.

IV.2 Syndex

Syndex est un logiciel de CAO (Conception Assisté par Ordinateur) niveau système, fondé sur la méthodologie adéquation algorithme-architecture (AAA) pour le prototypage rapide et l'optimisation d'applications distribuées temps réel embarquées complexes. Il permet de spécifier à l'aide de graphes les algorithmes applicatifs et les architectures distribuées, de lancer des heuristiques d'optimisation de l'implantation des algorithmes sur les architectures conduisant à une simulation temporelle de l'exécution temps réel, et enfin de générer automatiquement des exécutifs distribués temps réel sans interbloquage et à faible surcoût.

CHAPITRE III

CONTROLE A DISTANCE

I.Introduction

Le contrôle à distance selon l'explication anglaise du terme (remote control) est le contrôle et la commande d'un robot ou d'une machine quelconque à distance. C'est une technologie en plein épanouissement que ce soit sur le plan local (réseaux locaux) ou pour des réseaux non déterministe comme internet.
Le principe consiste à déplacer les commandes distante, pour que l'opérateur puisse avoir les informations à distance (images, localisation, retours d'efforts,....), cette distance varie de quelques mètres a plusieurs kilomètres, la liaison peut être de tout type (câble, fibre optique, micro-ondes).

II. Le concept de contrôle à distance [5,2]

Dans notre cadre, le robot manipulateur mobile a pour but de fournir à un agent humain un outil potentiellement mobile qui lui permette d'intervenir à distance sur un milieu particulier afin de remplir un objectif donné. Cette définition permet d'associer à cette branche de la robotique de nombreuse d'applications. De ces applications nous pouvons citer l'exploration dans des milieux inconnus, la surveillance industrielle, et l'intervention dans des milieux hostiles.

L'utilisation de robots manipulateur mobile à distance découle du besoin à un moment précis d'agir sur un environnement sans pouvoir (ou vouloir) être présent physiquement. Ils découlent donc tous de la volonté de fournir à un opérateur une sorte d'extension de son propre corps lui permettant d'agir à distance. La façon dont est réalisée cette commande à distance va elle aussi être amenée à varier. La distance séparant le robot de l'opérateur, le médium de communication utilisable, la quantité d'information échangeable ou encore la complexité du milieu dans lequel intervient le robot vont amener à répartir différemment les rôles à assumer par l'opérateur et par le robot. Entre deux usages d'un robot d'intervention, la succession d'actions entreprises, le site d'intervention, les difficultés rencontrées vont changer. Le robot et l'opérateur doivent s'adapter aux conditions de chaque opération à exécuter.

Dans le domaine de la commande à distance nous pouvons distinguer différents types qui peuvent être classifiés selon deux critères :

Premièrement suivant la nature de liaison entre le robot et l'opérateur :

- Liaisons mécaniques.

  Cette première méthode et sans doute la plus « ancienne » consiste à transmettre les ordres de l'opérateur au robot au moyen d'une liaison mécanique permettant de reproduire le mouvement souhaité. Cependant, cette mécanique de liaison limite fortement la distance à laquelle l'opérateur peut se trouver.

- Transmission électrique filaire :

  Dans ce genre de commande à distance, on supprime la liaison mécanique pour la remplacer par un médium véhiculant ordres, consignes ou tensions pour la commande du robot. On peut dès lors augmenter considérablement la distance entre l'opérateur et le robot.

- Transmission sans fil :

  Ce mode regroupe l'ensemble des transmissions non filaires disponibles. Même si la portée varie en fonction du moyen de communication utilisé (radio fréquences, infra-rouges ou ondes acoustiques), ce mode de transmission permet de s'affranchir de l'entrave que représente le lien filaire. Le robot n'est donc plus lié physiquement au poste de contrôle. Cette solution est une alternative au lien filaire comme par exemple avec les engins d'exploration spatiaux, lorsque l'on souhaite une grande mobilité du robot (domaine accessible plus important, moins de contraintes sur la trajectoire). Cela soulève toutefois des problèmes de débit d'information, de portée, de brouillage et de perte de contrôle du robot lorsque la communication est masquée ou perdue.

- Commande via un mode de transmission mixte filaire et non filaire :

  Ce dernier type de commande s'est considérablement développé avec l'apparition du réseau international qu'est Internet. L'idée de pouvoir commander un robot à partir de n'importe quel point de la « toile » ouvre des perspectives intéressantes en termes de ressources partagées et de souplesse d'utilisation. Le principe consiste à connecter l'opérateur au robot en passant d'abord par le réseau Internet puis par une borne émettrice (radio par exemple) communiquant avec le robot. On allie ainsi les avantages d'une liaison non filaire de courte ou moyenne portée (peu onéreuse et transportable) avec une grande distance entre l'opérateur et le robot. Cependant, le fait de passer par un réseau dont l'accès est non déterministe et dont la circulation de l'information peut varier dans l'espace (en fonction de l'itinéraire choisi par les routeurs) et dans le temps (en fonction de l'encombrement du réseau) présente des difficultés supplémentaires pour la commande à distance. En effet, un réseau comme Internet génère des retards variables et potentiellement des ruptures de communication, qui sont incompatibles avec une commande à distance "classique" consistant à faire parvenir au robot les commandes qu'il doit appliquer à ses actionneurs. Les retards dans le sens opérateur-robot entraînent une action décalée dans le temps, source d'instabilité d'autant plus sensible si le contrôle sur le robot est effectué en vitesse ou en effort. Dans l'autre sens, c'est la réaction de l'opérateur qui va être liée avec une situation qui n'est pas celle que rencontre le robot à l'instant présent. De même, si le robot doit faire face à l'apparition soudaine d'un obstacle sur sa trajectoire (un trou détecté au dernier instant par exemple), il faut être capable de réagir en un temps fini qui est incompatible avec les retards variables générés par Internet.

Dans nos travaux nous avons considéré ce mode de commande à distance. Malheureusement le réseau du CDTA souffrant d'une certaine saturation nous provoque des retards considérables. Pour palier à ce problème nous avons opté pour un réseau local entre le robot et l'opérateur distant.

Une deuxième classification de la commande à distance est possible en considérant le mode utilisé. Ce mode correspond au type d'interaction qui existe entre l'opérateur et le robot. Selon le mode d'interaction, le rôle de l'opérateur au sein de la commande et le niveau d'autonomie du robot sont différents. Les figures 1 à 3 illustrent l'ensemble des modes de commande à distance en présentant ces derniers suivant un ordre croissant quant à l'autonomie opérationnelle et décisionnelle dont est doté le contrôleur embarqué du robot.

- Robot non autonome :

  La figure 1a) présente le type de commande à distance de plus bas niveau. L'ensemble du contrôle du robot se trouve déporté du coté de l'opérateur. Le robot est alors défini comme non autonome puisqu'il est entièrement dépendant de la communication qui fait partie intégrante de la boucle d'asservissement. L'impact des retards de communication est alors directement répercuté sur la stabilité de la commande du robot. Dans la commande à distance de type consigne (figure 1b)), le robot acquiert un peu d'autonomie opérationnelle en embarquant les asservissements qui lui permettent de générer la commande.

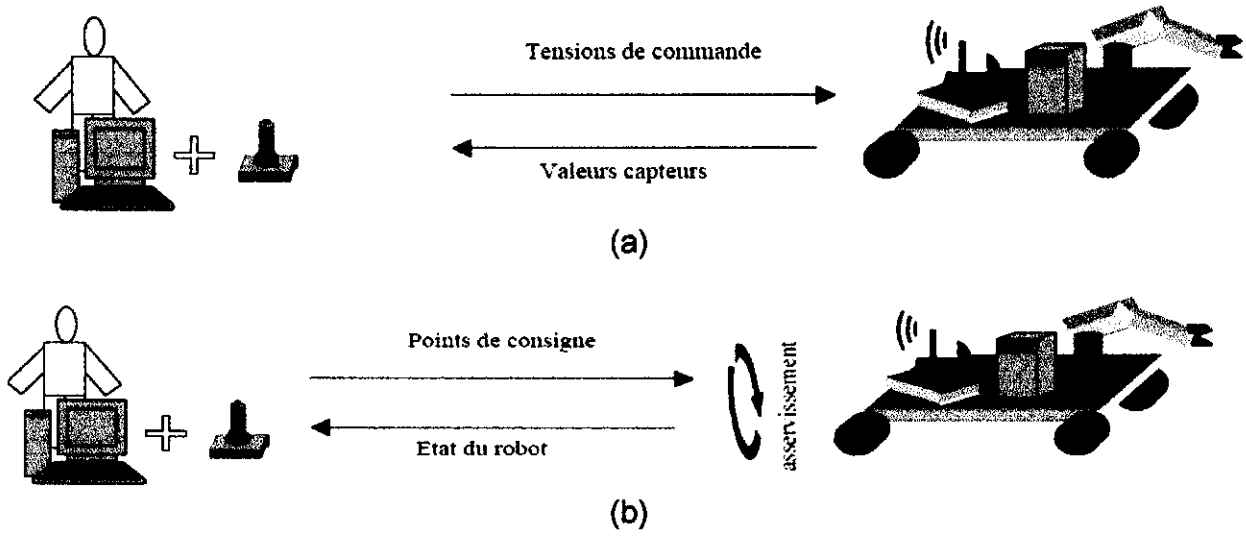


Figure 1. Commande à distance bas niveau

Des retards dans la transmission des consignes au robot n'occasionnent pas forcement une instabilité du robot. Cependant, le robot ne dispose tout de même que d'une autonomie plus que limitée. Là encore, le robot ne possédant aucune autonomie décisionnelle, il ne réalise aucune supervision.

- Robot à autonomie opérationnelle

La figure 2 présente une commande à distance de type tâche. Le robot dispose en plus des asservissements, de la possibilité de générer lui-même ses trajectoires. Il gagne encore en autonomie opérationnelle puisqu'il assume de plus en plus d'aspects dans la construction de sa commande. L'opérateur dispose donc de la totalité des décisions à prendre ce qui soumet toujours le robot aux aléas de communication.

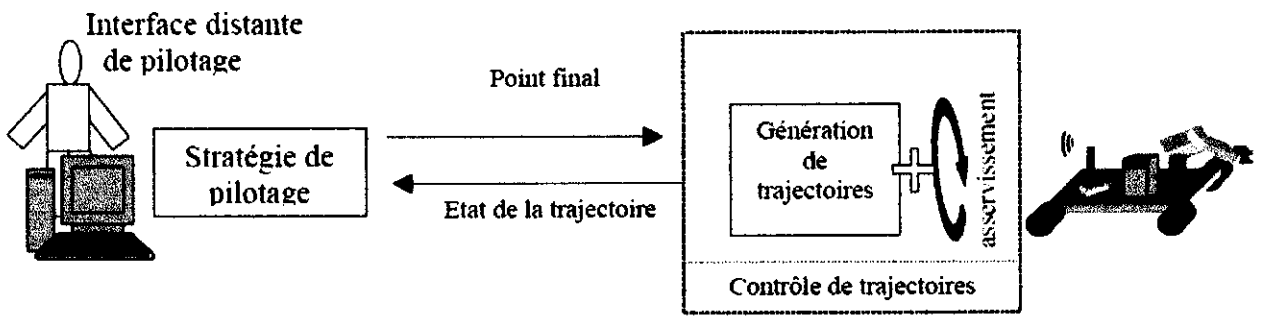


Figure 2 : Commande à distance à autonomie opérationnelle

- Robots à autonomie décisionnelle :

Les figures 3 a) et b) présentent les deux derniers types de commande à distance avant le robot entièrement autonome où l'opérateur n'intervient plus dans la boucle de contrôle. Le premier mode dit de type objectif permet de donner au robot des directives quant à ce qu'il doit accomplir. Le contrôleur embarqué « évolué » que nous appelons superviseur d'objectif permet de gérer et coordonner l'ensemble des ressources du robot afin de remplir un objectif donné. Cette séquence d'opérations peut être adaptée en fonction de paramètres liés à la situation du robot et à la perception qu'il a de son environnement. Il dispose dès lors d'un pouvoir décisionnel lui permettant de choisir sa séquence d'actions. L'opérateur n'intervient que pour transmettre les différents objectifs au robot, ou en tant que recours dans les situations où le robot ne trouve pas de solution. L'opérateur est indispensable dans ce schéma où l'intelligence embarquée reste limitée.

Le dernier type de commande à distance accroît encore le pouvoir décisionnel du robot. En effet, l'opérateur se contente dans ce dernier cas de fournir au robot une mission de façon très agrégée telle que "trouver un objet" ou "effectuer un prélèvement à tel endroit". En plus de son superviseur d'objectif, le robot dispose d'un planificateur qui va construire une séquence de sous-objectifs qui vont lui permettre d'accomplir sa mission. Ici, l'impact de la communication est minime et le robot peut parfaitement se trouver coupé de

l'opérateur pendant un certain temps sans que l'accomplissement de sa mission s'en ressente.

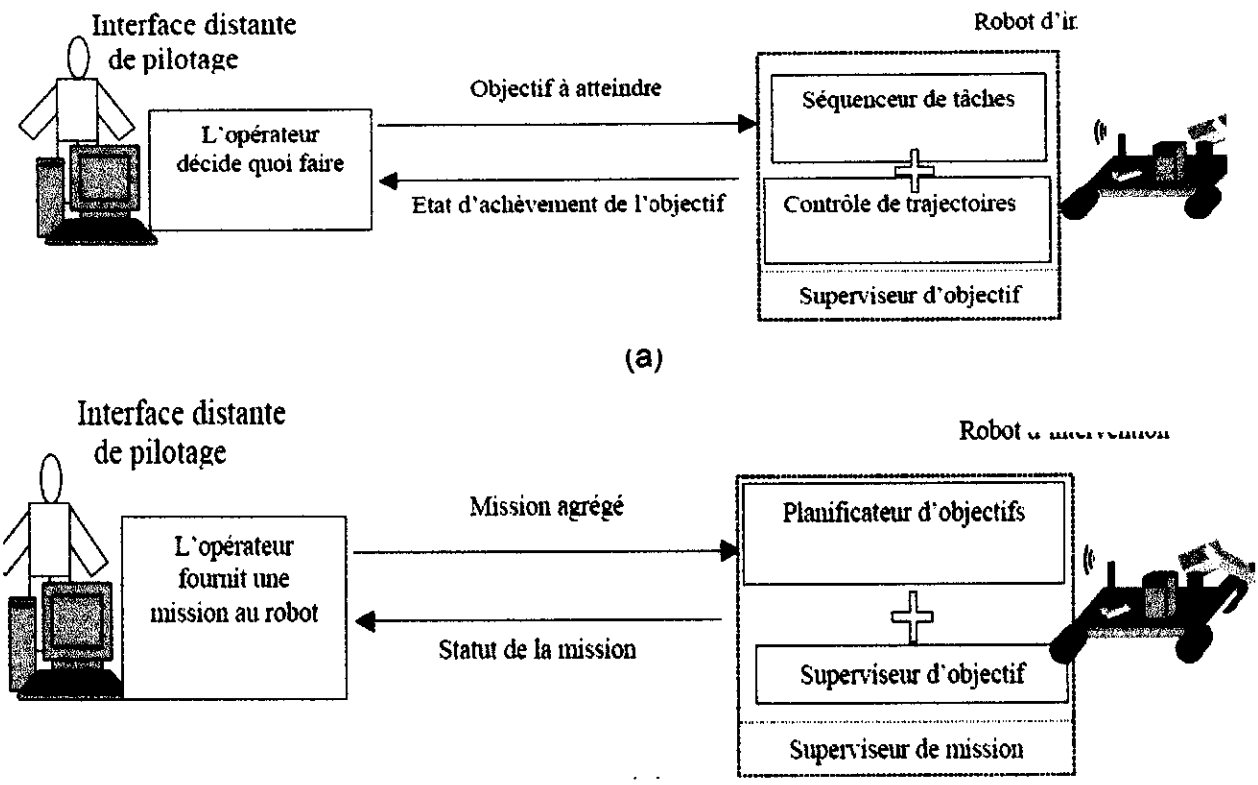


Figure 3 : Commande à distance à autonomie décisionnelle

Comme le robot du CDTA a été fourni totalement ouvert et vierge, nous nous consacrerons ici à une commande à distance de type bas niveau. Ce type de commande à distance n'exclue pas toutefois la possibilité de réaliser des commandes à autonomie opérationnelle. Cela peut constituer les travaux futurs sur ce site. Rien n'empêche, en effet, la coexistence de plusieurs modes avec passage (commutation) automatique de mode de commande à distance directe vers un mode autonome dès lors qu'un événement ne permettant plus la commande à distance directe survient (retard important). Le robot assume alors seul sa mise en situation stable et sûre. Cette commutation automatique de mode de commande à distance n'est toutefois pas traitée dans ce document.

Dans la suit du paragraphe nous présentons les outils informatiques qui nous permettent l'implémentation d'une commande à distance du robot robuter-ULM.

III Introduction aux réseaux [6,7]

Un réseau est un ensemble d'éléments ou d'objets interconnectes entres eux, qui permettent la circulation d'information entre eux. On peu parler de plusieurs réseaux: De réseaux téléphoniques, d'un réseau hydraulique pour les canalisations d'eaux, d'un réseau informatique,...etc.

Pour nous ce qui nous intéresse c'est les réseaux informatiques, un réseau informatique est un ensemble d'ordinateurs relies entre eux par des liens physiques (câble coaxial, fibre optique...) ou par des liaisons sans fil, et qui permettent l'échange de données entre eux (sous forme d'impulsions électrique, de lumière, ou d'ondes électro-magnetique pour les réseaux sans fil).

Les réseaux qui permettaient à leurs origine de relier des terminaux passifs aux ordinateurs centraux, permettent actuellement de relier tout type d'ordinateur (gros serveurs, stations de travail, terminaux ...).il sont donc indispensables pour les entreprises et les administrations (gestion, commerce, base de données, recherche), ainsi que pour les particuliers (Internet, jeux en réseau).

On distingue plusieurs types de réseaux:

- Réseau local (LAN) : qui peut s'étendre de quelques mètres a quelques kilomètres et qui satisfait les besoins internes des entreprises.

- Réseau métropolitain (MAN) : qui relie plusieurs sites situés dans la même ville, chacun possédant son propre réseau local, par exemple les différents sites d'une administration.

- Réseau étendu (WAN) : qui permet de communiquer a l'échelle d'un pays ou du monde entier, comme Internet, les infrastructure pouvant être terrestres ou spatiales à l'aide des satellites de télécommunications.

On distingue aussi plusieurs topologies de réseau : les réseaux en étoile, en anneau, en bus simple, en boucles et tout autre forme.

Il existe deux modes de réseaux : le mode point à point et le mode diffusion.

- Pour le mode point à point, le support relie une paire d'équipement seulement, les éléments non directement connectés le font par l'intermédiaire des autres noeuds du réseau.

- Pour le mode diffusion, tous les élément partagent le même support de transmission.Chaque donnée envoyée par un élément est reçue par tous les autres éléments, le destinataire reconnaîtra le message lui est destine grâce a l'adresse qu'il véhicule, un seul élément peut envoyer une donnée a un moment donné, c'est pour ça qu'il faut que l'émetteur écoute au préalable si la voie est libre. Dans une telle configuration, la rupture du support entraîne l'arrêt du

réseau, par contre l'arrêt d'une machine n'influe pas les autres, ce mode est très utilise dans les réseaux locaux.

On ne peut parler de réseaux informatiques sans parler de son plus grand problème qui est l'interconnexion des réseaux différents , au début des années 70, chaque constructeur apportait sa propre solution réseau (SNA d'IBM, DSA de Bull, TCP/IP..), il était impossible d'interconnecter ces réseaux si une norme n'était pas faite , et c'est pour ça que l'ISO (International Standard Organisation) a établie la norme OSI (Open Système Interconnections).Cela permet a tout système ouvert que se soit un ordinateur, un terminal ou un réseau respectant cette norme d'échanger des informations avec des équipements différents issus d'autres marques respectant cette norme aussi.

L'un des rôles essentiels de la norme était de définir un modèle pour toute les architectures, basé sur un découpage en sept couches, chacune ayant une fonction particulière.

7 Application
6 Présentation
5 Session
4 Transport
3 Réseau
2 Liaison
1 Physique

Figure 4 : Les sept couches de référence du modèle OSI de l'ISO

Chaque couche est constituée d'éléments matériels et logiciels et offre un service à la couche située immédiatement au-dessus d'elle en lui épargnant les détails de l'implémentation nécessaires. Chaque couche "n" gère la communication avec la même couche "n" de la machine distante en suivant un protocole bien spécifique.

- La couche physique : elle fournit les moyens mécaniques, électriques, fonctionnels et procéduraux nécessaires a l'activation, au maintien et à la désactivation des connexions physiques destinés à la transmission de bits entre deux entités de liaison de données.

- La couche liaison : elle fournit les moyens fonctionnels et procéduraux nécessaires à l'établissement, au maintien et à la libération des connexions de liaison de données entre entités du réseau. Elle détecte et corrige, si possible, les erreurs dues au support physique et signale au réseau les erreurs irrécupérables. Elle supervise le fonctionnement de la transmission et définit la structure syntaxique des messages, la manière d'enchaîner les échanges selon un protocole normalisé ou non.

- La couche réseau: elle assure toutes les fonctionnalités de relaie et d'amélioration de services entre entités de réseau, à savoir: l'adressage, le routage, le contrôle de flux et la détection et correction d'erreurs non réglées par la couche liaison.

- La couche transport: elle assure un transfert transparent entre entités de session, en les déchargeant des détails d'exécution. Elle a pour rôle d'optimiser l'utilisation des services du réseau disponible afin d'assurer au moindre coût les performances requises par la couche session.

- La couche session: elle fournit aux entités de la couche présentation les moyens d'organier et de synchroniser les dialogues et les échanges de données.

- La couche présentation: elle s'occupe de la syntaxe et de le sémantique des informations transportés, en se chargeant notamment de la représentation des données.

- La couche application: la couche application donne au processus d'application le moyen d'accéder à l'environnement OSI et fournit les services directement utilisables par l'application, qui sont le transfert de données, l'allocation de ressource, l'intégrité et la cohérence des données accédés et la synchronisation des application coopérantes.

IV. Introduction aux protocoles [8]

Un protocole est une méthode standard permettant la communication entre deux machines, c'est-à-dire un ensemble de règles et de procédures à respecter pour émettre et recevoir des données sur un réseau. Il en existe plusieurs selon nos attentes de la communication. Certains protocoles sont par exemple, spécialisés dans l'échange de fichiers (le FTP), d'autres peuvent servir à gérer simplement l'état de la transmission et des erreurs (c'est le cas du protocole ICMP),...

Sur Internet, les protocoles utilisés font partie d'une suite de protocoles, c'est-à-dire un ensemble de protocoles reliés entre-eux. Cette suite de protocoles s'appelle TCP/IP.
Elle contient, entre autres, les protocoles suivants:

-HTTP ;
-FTP ;
-ICMP ;
-TCP ;
-UDP ;
-SMTP ;
-NNTP ;
-Telnet ...

On classe généralement les protocoles en deux catégories selon le niveau de contrôle des données que l'on désire:

- Les protocoles orientés connexion: Il s'agit des protocoles opérant un contrôle de transmission des données pendant l'établissement d'une communication machines. Dans un tel schéma, la machine réceptrice envoie des accusés de réception lors de la communication, ainsi la machine émettrice est garantie de la validité des données qu'elle envoie. Les données sont ainsi envoyées sous forme de flot. TCP est un protocole orienté connexion.

- Les protocoles non orientés connexion: Il s'agit d'un mode de communication dans lequel la machine émettrice envoie des données sans prévenir la machine réceptrice, et la machine réceptrice reçoit les données sans envoyer d'accusé de réception à la première. Les données sont ainsi envoyées sous forme de blocs (datagrammes). UDP est un protocole non orienté connexion.

V. Le protocole TCP/IP [7, 8,9]

V.1. Introduction

TCP/IP est née de la réflexion de chercheurs américains suite à un problème posé par l'armée américaine , celle-ci disposait de plusieurs bases sur le territoire, chacune disposant de sa propre logistique informatique, donc de différentes machines reliées entre elle par des réseaux locaux, étant donnée que ces bases était reliées entre elle par des câbles, le problème était de trouver un moyen pour que l'information puisse circuler en reconfigurant le système automatiquement en cas de ruptures de liaison pour retrouver le chemin adéquat. De là, est né le protocole IP (Internet Protocole ou Interconnected Network Protocol).

IP est un protocole de la couche 3 du modèle OSI (la couche réseau), il assure sans connexion non fiable la délivrance de datagramme IP, et qui a pour fonction essentielle le routage, la définition du format des datagrammes ip qui est l'unité de base des données circulant sur le réseau, et la définition de la gestion de la remise non fiable des datagrammes.

Le problème de ce protocole est qu'il envoi l'information d'une machine à l'autre alors que l'information s'échange d'une application a l'autre, et c'est pour ça qu'on a crée le protocole TCP (Transport Control Protocol).

TCP est l'un des principaux protocoles de la couche transport (couche 6) du modèle OSI, il gère les données provenant ou à destination de la couche inférieure (le protocole IP) au niveau des applications. C'est un protocole orienté connexion qui permet à deux machine de communiquer en contrôlant l'état de la transmission, il assure un service fiable. Grâce à ce protocole la couche Internet ne se préoccupe que de l'envoie de données sous forme de datagramme sans se préoccuper du contrôle de données qui est assuré par le protocole TCP.

Donc le nom TCP/IP a été choisi en référence à ces deux principaux protocoles qui le caractérisent. Aujourd'hui TCP/IP intègre beaucoup d'autre protocole (ICMP, IGP, FTP, SMTP, HTTP, ...).

TCP/IP est très répandu à cause de robustesse prouvée (quelques millions de machines interconnectées dans le monde). Il est également très répandu, car dès son origine, il a été implémenté sur des systèmes Unix. Beaucoup de chercheurs ayant contribué à l'évolution de TCP/IP à son origine sont issus de l'université de Berkeley qui a très largement diffusé son système Unix avec l'interface des sockets pour manipuler des connexions TCP/IP.

V.2. Le protocole TCP

TCP est un protocole de la couche 6 (application) du modèle OSI, qui permet la gestion des données au niveau application, c'est un protocole orienté connexion, c'est-à-dire que l'état de la transmission est contrôlé. Les applications dialoguant lors d'une connexion sont considérées l'une comme un serveur et l'autre comme un client, qui doivent établir la connexion avant de pouvoir dialoguer. Après l'établissement de la connexion, les deux machines s'échangent des messages spécifiques. Cette connexion est bidirectionnelle simultanée (full duplex) composée de deux flots de données indépendants et de sens contraires. Les données sont encapsulées c'est à dire qu'on ajoute aux paquets de données un entête qui va permettre la synchronisation de la transmission et d'assurer leurs réception.

TCP échange entre les deux machines un flux d'octet non interprété par TCP, c'est aux applications des deux extrémités que revient ce travail.
Dans le cas d'informations trop volumineuses, elles sont fractionnées en données de taille optimale par TCP. De même TCP peut regrouper des données pour former qu'un seul datagramme de taille convenable pour décharger le réseau, cette unité est appelée segment. La figure () montre le format des données TCP.

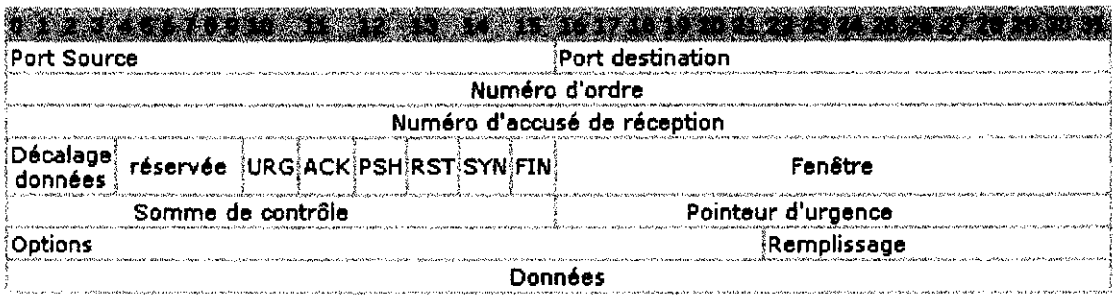


Figure5 : Format des données TCP

Tableau 1 : Explication des différents champs d'une données TCP

Port Source	16 bits	Port relatif à l'application en cours sur la machine source.
Port Destination	16 bits	Port relatif à l'application en cours sur la machine de destination.
Numéro d'ordre	16 bits	Lorsque le drapeau SYN est à 0, le numéro d'ordre est celui du premier mot du segment en cours Lorsque SYN est à 1, le numéro de séquence est le numéro de séquence initial utilisé pour synchroniser les numéros de séquence (ISN).
Numéro d'accusé de réception	32 bits	Dernier segment reçu par le récepteur.
Décalage des données	4 bits	Il permet de repérer le début des données dans le paquet. Le décalage est ici essentiel car le champ d'options est de taille variable.
Réservé	6 bits	Champ inutilisé actuellement mais prévu pour l'avenir.
Drapeaux (oflags)	6x1 bits	Les drapeaux représentent des informations supplémentaires: URG: si ce drapeau est à 1 le paquet doit être traité de façon urgente ACK: si ce drapeau est à 1 le paquet est un accusé de réception PSH (PUSH): si ce drapeau est à 1, le paquet fonctionne suivant la méthode PUSH RST: si ce drapeau est à 1, la connexion est réinitialisée SYN: si ce drapeau est à 1, les numéros d'ordre sont synchronisés FIN: si ce drapeau est à 1 la connexion s'interrompt
Fenêtre	16 bits	Champ permettant de connaître le nombre d'octets que le récepteur souhaite recevoir sans accusé de réception.
Somme de contrôle	Checksum ou CRC	La somme de contrôle est réalisée en faisant la somme des champs de données de l'en-tête, afin de pouvoir vérifier l'intégrité de l'en-tête.
Pointeur d'urgence	16 bits	Indique le numéro d'ordre à partir duquel l'information devient urgente.
Options	Taille variable	Des options diverses.

Remplissage		On remplit l'espace restant après les options avec des zéros pour avoir une longueur de 32 bits.

Les principales caractéristiques du protocole TCP sont:

- Permet de remettre en ordre les datagrammes en provenance du protocole IP ;
- Permet de vérifier le flot de données afin d'éviter une saturation du réseau ;
- Permet l'initialisation et la fin d'une communication de manière courtoise.
- Permet de formater les données en segments de longueur variable afin de les remettre au protocole IP.

Il a aussi deux très importantes caractéristiques qui sont:

-La fonction de multiplexage :

C'est faire transiter sur une même ligne des données provenant d'applications diverses. Ces opérations sont réalisées grâce au concept de ports (ou sockets), c'est-à-dire un numéro associé à un type d'application, qui, combiné à une adresse IP, permet de déterminer de façon unique une application qui tourne sur une machine donnée.

-Fiabilité des transferts :

TCP est un protocole fiable. Il possède un système d'accusé de réception qui assure la bonne réception des données. Lors de l'émission d'un segment, un numéro de séquence lui est associé. A la réception de ce segment, la machine réceptrice va renvoyer un accusé de réception (un segment de donnée dont le drapeau ACK est à 1) munis d'un numéro égal au numéro de la séquence précédant.

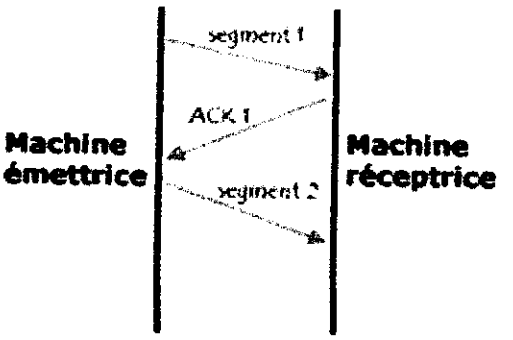


Figure 6 : Fiabilité des transferts

Aussi à chaque envoi d'un segment, la machine émettrice enclenche une minuterie, qui si elle expire et que l'accusé de réception n'arrive pas, alors dans ce cas la machine émettrice considère que le segment est perdu et l'envoi de nouveaux.

Toutefois, si le segment perdu arrive à destination en retard, alors la machine réceptrice saura que c'est un doublon grâce à son numéro de séquence, et l'éliminera.

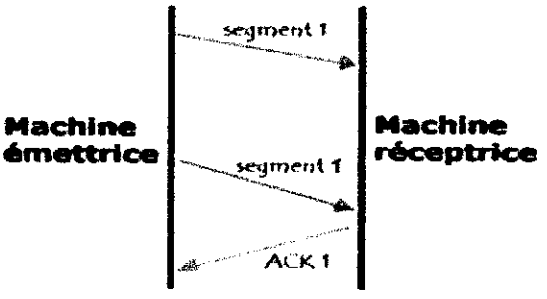


Figure 7 : Fiabilité des transferts

V.3 La connexion TCP

Etablissement d'une connexion :

Etant donné que ce processus de communication, qui se fait grâce à une émission de données et d'un accusé de réception, est basé sur un numéro d'ordre (appelé généralement numéro de séquence), il faut que les machines émettrices et réceptrices (client et serveur) connaissent le numéro de séquence initial de l'autre machine.

L'établissement de la connexion entre deux applications se fait souvent selon le schéma suivant:

- Les ports TCP doivent être ouverts ;
- L'application sur le serveur est passive, c'est-à-dire que l'application est à l'écoute, en attente d'une connexion ;
- L'application sur le client envoie une requête de connexion au serveur dont l'application est en ouverture passive. L'application du client est dite « en ouverture passive ».

- Les deux machines doivent donc synchroniser leurs séquences grâce à un mécanisme commun appelé three ways handshake (poignée de main en trois temps), que l'on retrouve aussi lors de la clôture de session.

Ce dialogue permet d'initier la communication, il se déroule en trois temps, comme sa dénomination l'indique:

- Dans un premier temps, la machine émettrice (le client) transmet un segment dont le drapeau SYN est à 1 (pour signaler qu'il s'agit d'un segment de synchronisation), avec un numéro d'ordre N, que l'on appelle numéro d'ordre initial du client.

- Dans un second temps, la machine réceptrice (le serveur) reçoit le segment initial provenant du client, puis lui envoie un accusé de réception, c'est-à-dire un segment dont le drapeau ACK est à 1 et le drapeau SYN est à 1 (car il s'agit là encore d'une synchronisation). Ce segment contient le numéro d'ordre de cette machine (du serveur) qui est le numéro d'ordre initial du client. Le champ le plus important de ce segment est le champ accusé de réception qui contient le numéro d'ordre initial du client, incrémenté de 1.

- Enfin, le client transmet au serveur un accusé de réception, c'est-à-dire un segment dont le drapeau ACK est à 1, et le drapeau SYN est à zéro (il ne s'agit plus d'un segment de synchronisation). Son numéro d'ordre est incrémenté et le numéro d'accusé de réception représente le numéro de séquence initial du serveur incrémenté de 1.

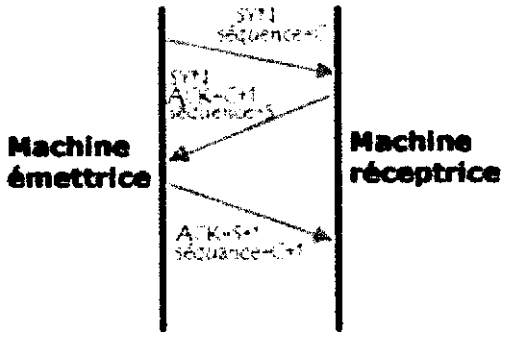


Figure 8 : Synchronisation de deux machines.

Suite à cette séquence comportant trois échanges, les deux machines sont synchronisées et la communication peut commencer.

Fin d'une connexion :

Le client peut demander à mettre fin à une connexion au même titre que le serveur.
La fin de la connexion se fait de la manière suivante:

- Une des machines envoie un segment avec le drapeau FIN à 1, et l'application se met en état d'attente de fin, c'est-à-dire qu'elle finit de recevoir le segment en cours et ignore les suivants ;
- Après réception de ce segment, l'autre machine envoie un accusé de réception avec le drapeau FIN à 1 et continue d'expédier les segments en cours.
- Suite à cela, la machine informe l'application qu'un segment FIN a été reçu, puis envoie un segment FIN à l'autre machine, ce qui clôture la connexion....

V.4. Adressage

Chaque ordinateur d'un réseau Internet possède une adresse IP unique codée sur 32 bits. Une adresse est représenté de 4 nombres séparés par des points dans une "notation décimale pointe" avec un octet pour chaque nombre, chaque nombre est compris entre 0 et 255. Une adresse IP est constitue d'une paire (id. de réseau, id. de machine) et appartient a une certaine classe (A, B, C, D et E) selon la valeur de son premier octet.

Donc on a pour le premier octet de :

- 0 a 127 : classe A
- 128 a 191 : classe B
- 192 a 223 : clase C
- 224 a 239 : classe D
- 240 a 247 : classe E

Les adresses de classe A sont utilisées pour les très grands réseaux qui comporte plus de 65536 ordinateur comme celui du pentagone ou du MIT, le nombre de réseaux de classe A est limite a 127 dans le monde.

Les adresses de classe B sont réservées aux réseaux ayant entre 256 et 65536 ordinateurs.

Pour la classe C on ne peut dépasser les 256 machines, le nombre de réseaux de ce type peut dépasser 2 millions.

Certaines adresses IP ont une signification particulières on peut citer :

- 0.0.0.0 utilise par une machine pour connaître sa propre adresse IP.
- <id.de réseau>.<id.de machine nul> permet de désigne le réseau lui-même.
- <id.de réseau>.<id.de machine avec tous ses bits a 1> est une adresse de diffusion (broadcasting)
- les adresses de classe a de 10.0.0.0 a 10.255.255.255, de classe b de 172.16.0.0 à 176.31.255.255 et de classe c de 192.168.0.0 a 192.168.255.255 sont réservées à la constitution de réseaux intranet.

Le système des adresse IP permet aussi la création de sous-reseaux en découpant la partie réservée à l'adresse machine sur un réseau en deux parties dont la première sera un identificateur de sous-reseaux.

VI. Le modèle client/serveur [6,10]

Le modèle client/serveur est apparu dans les année 90 de l'aboutissement d'un ensemble d' évolutions technologiques (capacité mémoire, performance des processeurs et des réseaux, évolution des logiciel : interface graphique, multimédia, interface de communication) , pour que tout utilisateur d'une entreprise ou autre puisse accéder à tout information autorisée par les règles de confidentialité et de sécurité, de

manière instantanée depuis n'importe quel poste en utilisant une interface aussi simple que possible.

Une architecture client/serveur signifie que des machines clientes (des machines faisant partie du réseau) contactent un serveur, une machine généralement très puissante, qui leur fournit des services. Ces services sont des programmes fournissant des données telles que l'heure, des fichiers, une connexion, ...

Cette architecture comporte plusieurs caractéristiques, on peut citer les plus importantes:

- Le serveur est fournisseur de service et le client est consommateur.
- C'est toujours le client qui déclenche la demande de service. Le serveur attend passivement la requête du client.
- Un serveur traite plusieurs clients au même temps et contrôle leur accès aux ressources.
- Il est possible d'ajouter et de retirer des stations clientes. Il est possible de faire évoluer les serveurs.
- Le modèle client-serveur est indépendant des plates-formes matérielles et logicielles.
- On peut modifier le serveur sans toucher au client. La réciproque est vraie.

Cette architecture est utilisée dans plusieurs domaines telle: la gestion de base de données, systèmes transactionnels, système de messagerie, web, Internet....etc, vu tous les avantages qu'elle comporte (des ressources centralisées: étant donné que le serveur gère des ressources communes à tous les utilisateurs, une meilleure sécurité: car le nombre de points d'entrée permettant l'accès aux données est moins important, une administration au niveau serveur , un réseau évolutif ou l'on peut supprimer ou ajouter des clients sans perturber le réseau).

Néanmoins, cette architecture a quelque inconvénients telle que le coût élevé du à la technicité du serveur, et aussi ,le maillon faible que constitue le serveur étant donné que toutes l'architecture tourne autour de lui donc dépendante de lui.

La figure montre le fonctionnement d'une architecture client/serveur

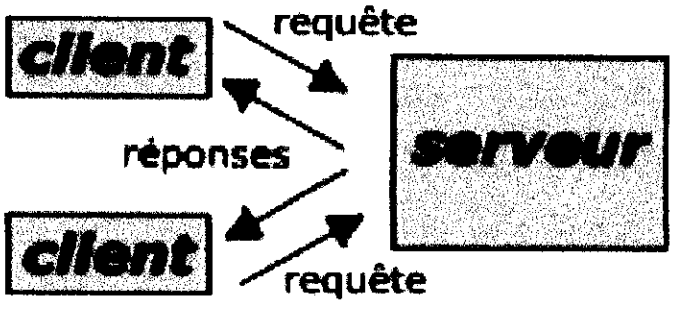


Figure 9 : Fonctionnement d'une architecture client/serveur

VI.1 les socket

On va parler maintenant d'un mécanisme important pour l'implémentation d'une architecture client/serveur qui est le socket.

Dans une connexion client/serveur chaque extrémité utilise une socket, on a une socket client et une autre socket serveur.

La socket est identifié grace à son adresse IP et à son numéro de port.

La socket client demande une connexion à une socket serveur.

La socket serveur attend une demande de connexion puis dialogue avec le ou les sockets clientes dont la requête est acceptée.

Quand un processus serveur gère chaque requête de façon indépendante, il est qualifié de serveur itératif. Il gère les requêtes une par une dans leurs ordre d'arrivée mais il n'y a pas de chevauchement entre les traitements de requêtes.

A l'inverse, quand on ne peut pas prévoir le temps nécessaire à un serveur pour répondre, on dit qu'il s'agit d'un serveur concurrent. Ce serveur crée un processus distinct pour chaque requête de service. Typiquement, à chaque fois qu'une connexion est établie, le serveur lance un nouveau processus qui reste en écoute d'éventuelles autres demandes de connexion. Cela nécessite un environnement multi-tâches.

Dans le cas du protocole TCP, ou l'on travaille en mode connecté, la figure montre une socket en mode connecte.

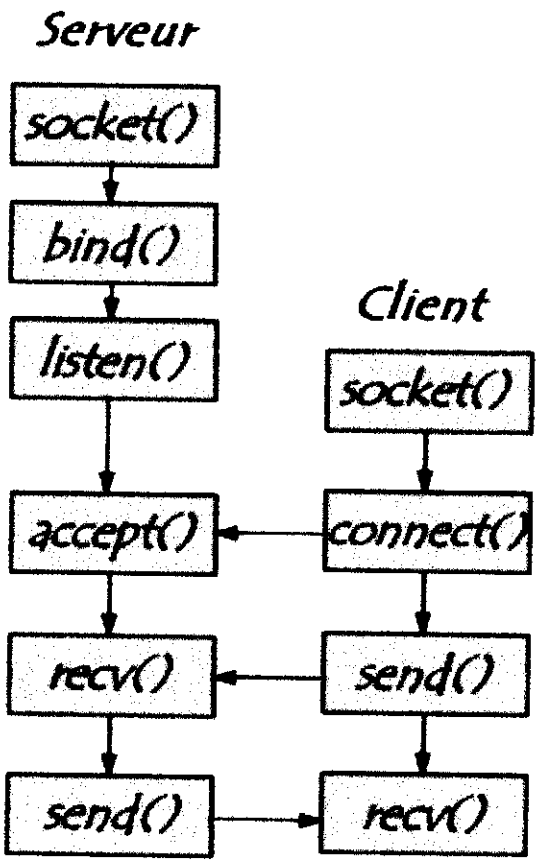


Figure 10 : Socket en mode connecté

Dan le cas du protocole UDP, ou l'on travail en mode deconnecté, la figure () montre une socket en mode deconnecté.

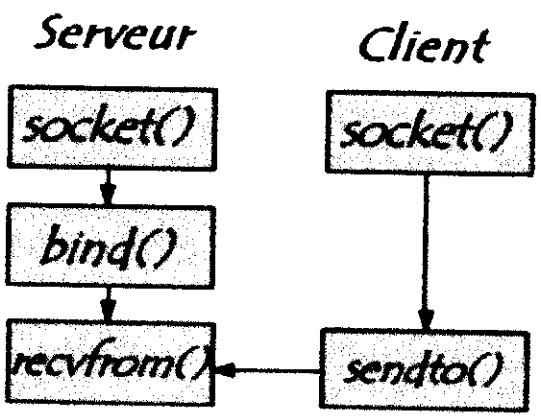


Figure 11 : Socket en mode deconnecté.

Voici les définitions des fonctions citées là-dessus qu'on verra plus en détail dans le chapitre 5:

Socket(): pour la création de la socket d'écoute.

Bind(): pour l'attribution du port d'écoute.

Listen(): pour l'écoute du port, et l'attente d'une éventuelle demande de connexion de la part d'un client.

Accep() : pour l'acceptation du client et la creation d'un socket de communication.

Recv() : pour la réception de données.

Send() : pour l'émission de données.

Connect() : pour la tentative de connexion au serveur de la part du client.

CHAPITRE IV

CONCEPTION DE L'APPLICATION LOGICIELLE

I. Méthodologie de l'application [11,12]

I.1 Historique d'UML

L'approche objet est depuis longtemps une réalité, les concepts de base sont stables est largement approuvés. L'approche objet a commencé à apparaître vers la fin des années 60 , avec Simula qui a été le premier langage de programmation à implémenter le concept de type abstrait a l'aide des classes, et en 1976 Smaltalk implémente les concepts fondateurs de la méthode objet : encapsulation , agrégation et héritage . Les premiers compilateurs C++ font leur apparution en début des années 80 et de nombreux langages oriente objet voient le jour (Eiffel, loops...).

Actuellement l'approche objet est devenue incontournable, dés lors qu'on cherche à concevoir des logiciels complexes qui doivent résister a des évolutions incessantes, la programmation objet bénéficie d'une panoplie d'outils et de langages performants.

Mais l'approche comporte quelques inconvénients, qui peuvent induirent leurs utilisateurs dans l'erreur et faire couler le projet.

De ces difficultés on peut citer :

- L'approche objet n'est pas très intuitive, car il est difficile pour le cerveau humain de décomposer un problème informatique en termes d'objet et d'interaction entre ces objets, et rien dans les concepts de base de l'approche objet ne dicte comment modéliser la structure objet d'un système de manières pertinentes.
- L'application des concepts objet nécessite beaucoup de rigueurs, car il y a beaucoup d'embiguités et d'incompréhensions, et cela est due au fait que beaucoup de développeur pensent souvent objet qu'a travers les langages de programmation, alors que ceux-ci ne valident en rien l'utilisation de moyens techniques pour concevoir un système conforme a la philosophie objet.

C'est pour cela qu'il il faut disposer d'un outil qui sera un guide dans l'utilisation des concepts objets.

Et pour cela il nous fau t:

1) un langage (pour s'exprimer clairement à l'aide des concepts objets), qui doit permettre de

- représenter des concepts abstraits (graphiquement par exemple),
- limiter les ambiguïtés (parler un langage commun, au vocabulaire précis, indépendant des langages de programmation orientés objet),
- faciliter l'analyse (simplifier la comparaison et l'évaluation de solutions).

2) une démarche d'analyse et de conception objet pour :

- Ne pas effectuer une analyse fonctionnelle et se contenter d'une implémentation objet, mais penser objet dès le départ.
- Définir les vues qui permettent de décrire tous les aspects d'un système avec des concepts objets.

C'est pour ça que les langages de modélisation orientés objets ont fait leur apparition, au milieu des années 70, quand les spécialistes ont commencé à expérimenter de nouvelles approches d'analyse et de conception. Entre 1989 et 1994 le nombre de méthodes objet est passé de 10 a 50, sans qu'elles répondent vraiment aux besoins des utilisateurs.

Basées sur l'expérience acquise, de nouvelles méthodes sont apparues, les plus importantes sont Booch, OOSE (Object Oriented Software Engineering) et OMT (Object Modeling Technique), elles ont été reconnues au niveau mondial comme étant incontournable. Chacune de ses méthodes constituait une méthodes complète mais présenter encore des inconvénients, en résumé chacune des méthodes avait un créneau dans lequel elle excellait.

Au milieu des années 90 les principaux auteur des méthodes, Booch, OOSE et OMT que sont respectivement Grady Booch (Rational Software Corporation), Ivar Jacobson (objectory) et James Rumbaugh (General Electric) ont commencé à adopter les idées des deux autres, puis se sont décidés à travailler ensemble pour la création d'un langage de modélisation unifié, et cela pour trois raisons :

- Poursuivre cette évolution ensemble pour éliminer les différences inutiles qui aurait embrouillé les idées des utilisateurs.
- Apporter une stabilité au marche orienté objet.
- Apporter des améliorations aux trois méthodes et répondre a des problèmes qu'aucune d'elle ne traiter de manière satisfaisante.

Et pour atteindre es objectifs suivants :

1-La modélisation des systèmes au moyens de techniques orienté objet, depuis leur conception jusqu'à leur artefact exécutable.

2-La résolution des problèmes d'échelles inhérentes aux systèmes complexes et essentiels.

3-La création d'un langage de modélisation utilisable à la fois par les humains que les machines.

De là, donc est née l'UML (Unified Modeling Language) traduit (langage de modélisation objet unifié) de la fusion des trois méthodes (Booch, OMT, OOSE), qui est le fruit d'un large consensus, de très nombreux acteurs industriels de renoms (HP,

IBM, ORACLE, MICROSOFT.....) l'ont adopte et participe à son développement .en un peu de temps UML est devenus un standard incontournable.

I.2 Description d'UML

UML (Unified Modeling Language) est un langage standard conçu pour l'écriture des plans d'élaboration de logiciel. Il peut être utilisé pour visualiser, spécifier, construire et documenter les artefacts d'un système à fortes composantes logicielles.

UML est utilisé pour visualiser dans le sens où il permet d'utiliser au mieux les représentations graphiques ou textuelles en fonction des besoins, où chaque symbole UML possède une sémantique bien définie, et chaque développeur ou outil peut interpréter ce modèle sans ambiguïté.

UML est un langage pour spécifier dans le sens où il construit des modèles précis, sans ambiguïté et complets. Et cela en spécifiant toutes les décisions importantes en termes d'analyse, de conception, et d'implémentation.

UML est un langage pour construire dans le sens où l'on peut construire ou traduire les modèles d'UML dans un langage de programmation tel que JAVA, C++..... Et cela s'appelle l'application de l'igenierie vers l'aval. L'inverse est aussi possible c'est-à-dire passer du code vers le modèle ou plus precesimment la reconstruction du modèle à partir du code. Cela s'appelle la retro-ingenierie, mais il nécessite le support d'un outil pour cause de perte d'informations.

UML est un langage pour la documentation dans le sens où il permet de documenter l'architecture d'un système dans ses moindres détails.

UML est utilisé dans beaucoup de domaine tel :

- Les services informatiques d'entreprise.
- Les services bancaires et financiers.
- Les télécommunications.
- Les transports.
- La défense et l'aero-spatiale.
- Le commerce de détail.
- L'électronique de détail.
- Les sciences.
- Les services distribués basé sur le web mais aussi à des systèmes qui n'appartiennent pas à cette catégorie, comme le workflow d'un système judiciaire, ou la conception de matériel informatique.

Les points forts d'UML:

UML est un langage formel et normalisé : gain de précision, gage de stabilité, encourage l'utilisation d'outils.

République Algérienne Démocratique et Populaire.
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique.

Université Saad Dahlab. Blida
USDB.

Faculté des sciences.
Département informatique.

Mémoire pour l'obtention
d'un diplôme d'ingénieur d'état en informatique.
Option IA

Sujet :

CONTROLE D'UN ROBOT MANIPULATEUR MOBILE BASE SUR UNE ARCHITECTURE CLIENT/SERVEUR

Présenté par : LAIB LAHCENE
BESSAIH REDOUANE

Promoteur : KADRI Mohamed
Encadreur : BOUCHEMMA Rafik

Thème proposé par :
Equipe Système Robotisé de Production
Division Robotique et Productique (CDTA)

Soutenue le: 02/10/2005, devant le jury composé de :

M. BALA	**Université Saad Dahlab, Blida USDB**	**Président**
M. MAHIEDDINE	**Université Saad Dahlab, Blida USDB**	**Examinateur**

PROMOTION 2004/2005

Dédicace de hcene

Je dédie premièrement ce mémoire à mes parents qui m'ont soutenu tout au long de ma vie, qui m'ont guidé et orienté dans le bonne voie .je les remercie pour m'avoir aidé à surmonter les obstacles.

Et aussi mes frères, et a mes cher amis Mazouzi Abdelghani,Rabie ,Samir ,Nabi, Hafid, Amine,l ainsi que tous les autres

Dédicace de Redouane

Je dédie premièrement ce mémoire à mes cher parent qui m'ont soutenu depuis toujours et dans toutes les épreuves.

Je le dédie aussi à mes frères et soeurs, a ma tante et a ma belle soeurs, ainsi que toutes ma famille de prés ou de loin.

Enfin je le dédie à mes amis Marzak, Boussaad, Bilal, Rouji et tous les autres.

SOMMAIRE

CHAPITRE I

INTRODUCTION GENERALE

INTRODUCTION GENERALE

Le domaine de la robotique a connue une croissance énorme lors des dernières années. Les avancées scientifiques, le perfectionnement des systèmes mécaniques et l'augmentation de l'intégration ont permis de passer de la génération des robots manipulateurs fixes à celle des robots mobiles, puis à celle des robots manipulateurs mobiles. Pour une meilleure exploitation de ces systèmes, pendant longtemps les efforts ont été dirigés vers leurs maîtrise du point de vue de la régulation qui permet au robot de suivre au plus exact un comportement (des trajectoires) strictement connu à l'avance. Les difficultés résident, d'une part, dans l'écriture d'un modèle réaliste, c'est-à-dire exact, du robot, d'autre part, dans la découverte du type de contrôle adéquat, sachant qu'on doit faire face à un système très non linéaire.

L'apport de ce développement technologique a permis à la robotique de s'élargir à de nombreux domaines comme la médecine, le militaire, l'exploration. Du fait de cet élargissement les robots sont amenés à intervenir dans des milieux très différents tels que le milieu sous-marin, aérien, spatial (robots sondes d'exploration) ou encore le milieu terrestre.

Même si Jusqu'à maintenant, les coûts élevés ont limité leur utilisation dans l'aérospatiale, dans les domaines militaires, et dans les centrales nucléaires. Néanmoins, la prolifération des développements dans ce domaine permet l'introduction de cette technologie dans les domaines commerciaux de l'agriculture, et aussi dans l'industrie, des services, des mines, de la médecine et d'autres encore. C'est un secteur stratégique avec un fort potentiel de croissance.

Le fait que ces robots interviennent dans des milieux aussi variés et de plus en plus dans des endroits inaccessibles à l'homme a soulevé de nombreux problèmes liés à la commande de ces robots. L'éloignement de l'opérateur humain a impliqué de développer des systèmes de perception et de communication de plus en plus évolués, de doter les robots de facultés de décision et d'analyse permettant de suppléer partiellement à la décision fournie par l'opérateur. Tout cela a abouti à la création de robots commandés à distance avec différents niveaux d'autonomie.

La commande d'une machine, d'un outil à distance rend souvent la vie de l'opérateur plus simple, plus pratique. L'homme à par exemple, depuis longtemps voulu commander un véhicule sans devoirs se trouver à l'intérieur. Dans l'absolu, la commande à distance intervient lorsque l'opérateur ne peut agir lui-même, pour des raisons de sécurité (endroits exposés à de la radioactivité), d'inaccessibilité (tuyaux d'aération), d'agressivité du milieu (espace, grandes profondeurs),...

La recherche en robotique s'est quelque peu réorientée par suite des échecs sur l'autonomie qui ont obligé à de nouvelles réflexions. De la nous pouvons dire qu'opter pour l'autonomie des robots serais très difficile a concevoir sans garantie de résultat, pour cela beaucoup ont opté pour le contrôle a distance du robot.

Les systèmes commandés par des réseaux informatiques sont un domaine en pleine effervescence au sein de la communauté scientifique vu l'apport de la nouvelle technologie (réseaux informatique, Internet) et les avancées en termes des performances des ordinateurs (processeurs, mémoires.....) rend plus facile le contrôle à distance des robots et l'exécution des tâches en temps réels. On peut ainsi contrôler des robots (inspection, déminage, sauvetage, médecine.) ou même des équipements lourd comme le télescope, le contrôle distant étant rendu possible par l'introduction d'un réseau dans la boucle de commande du système cybernétique. Tout cela induit des problèmes spécifiques tels:

-Ergonomie du poste de travail (élaboration de la consigne).

-Qualité des retours informationnels (réalité vertuelle et augmentée).

-Retard de transmission des informations (dus au réseau).

-Sécurité de l'application.

Face à la diversité, l'hétérogénéité et la complexité des systèmes à mettre en oeuvre, il est nécessaire de doter le robot d'une architecture de commande permettant de coordonner et de piloter l'ensemble. Les architectures de commande remplissent plusieurs fonctions. Elles permettent tout d'abord de conférer au robot un comportement spécifique permettant l'accomplissement de sa mission. Pour cela, elle met en oeuvre tout ou partie des équipements composant le robot en organisant et gérant les ressources de ce dernier. D'autre part, l'architecture octroie au robot une part d'autonomie fonctionnelle et/ou décisionnelle permettant à ce dernier d'interagir avec son environnement. Enfin, l'architecture de commande doit permettre de superviser les actions entreprises par le robot par l'opérateur avec les contraintes générées par l'environnement.

Parmi les travaux entrepris dans cet axe de recherche, nous pouvons citer les travaux de Arnaud Leleve [1] de l'université de Montpellier sur le télépilotage à longue distance d'engins mobiles robotises pour des tâches d'intervention en milieu difficile ou hostile pour l'homme (nucléaire, spatial, sous-marins,...). Ses travaux ont permis de mettre en place un système de communication entre un poste de télépilotage Assisté par Ordinateur et un engin mobile robotisé (manipulateur mobile constitué d'une base Andruet et d'un bras manipulateur de type Puma).

Vicente Egea and all [2] ont pu concevoir un véhicule autoguidé en utilisant sur une commande basée sur une architecture client/serveur pour permettre un contrôle à distance de leur véhicule.

Nous souhaitons réaliser une architecture de commande à distance pour un robot manipulateur mobile (Robuter-ULM du CDTA). Nous partons du postulat que le robot possède à l'intérieur de sa chaîne de contrôle un opérateur humain. La commande repose sur une architecture client/serveur, le serveur tourne sur le Pc embarqué du robot manipulateur mobile, et le client un Pc hôte doté d'une interface

utilisateur qui permet à l'opérateur de percevoir l'état du robot et d'envoyer ces consignes vers le serveur pour qu'ils soient exécuter par le robot.

Nous allons donc présenter dans le deuxième chapitre le site expérimental sur lequel nous comptant implémenter notre architecture de commande. Nous présenterons dans les détails les caractéristiques du robot manipulateur mobile (Robuter-ULM du CDTA)

Dans le troisième chapitre, le principe de la commande à distance sera exposé en plus des différents outils informatiques se reportant à l'architecture réseau utilisée.

Dans le quatrième chapitre, nous présenterons la méthodologie de conception de notre application en se basant sur le langage de modélisation UML.

Enfin, dans le dernier chapitre, nous mettrons en œuvre notre architecture de commande sur robot Robuter-ULM. L'application réalisée fera l'objet d'une présentation en illustrant ses diverses fonctionnalités.

Notre document se verra terminer par une conclusion générale.

CHAPITRE II

PRESENTATION DU ROBOT ROBUTER-ULM

I. Introduction

Le Robuter-ULM [3] montré en figure 1 est un robot manipulateur mobile constitué d'une base mobile à 4 roues et d'un bras manipulateur à six axes avec une pince électrique à son extrémité. Le robot est équipé d'une ceinture de capteurs ultrasons, d'une Camera CCD et d'un capteur d'effort. Le système est commandé par un Pc industriel embarqué MMX 233 et quatre cartes Robosoft à base de microcontrôleur MPC555.

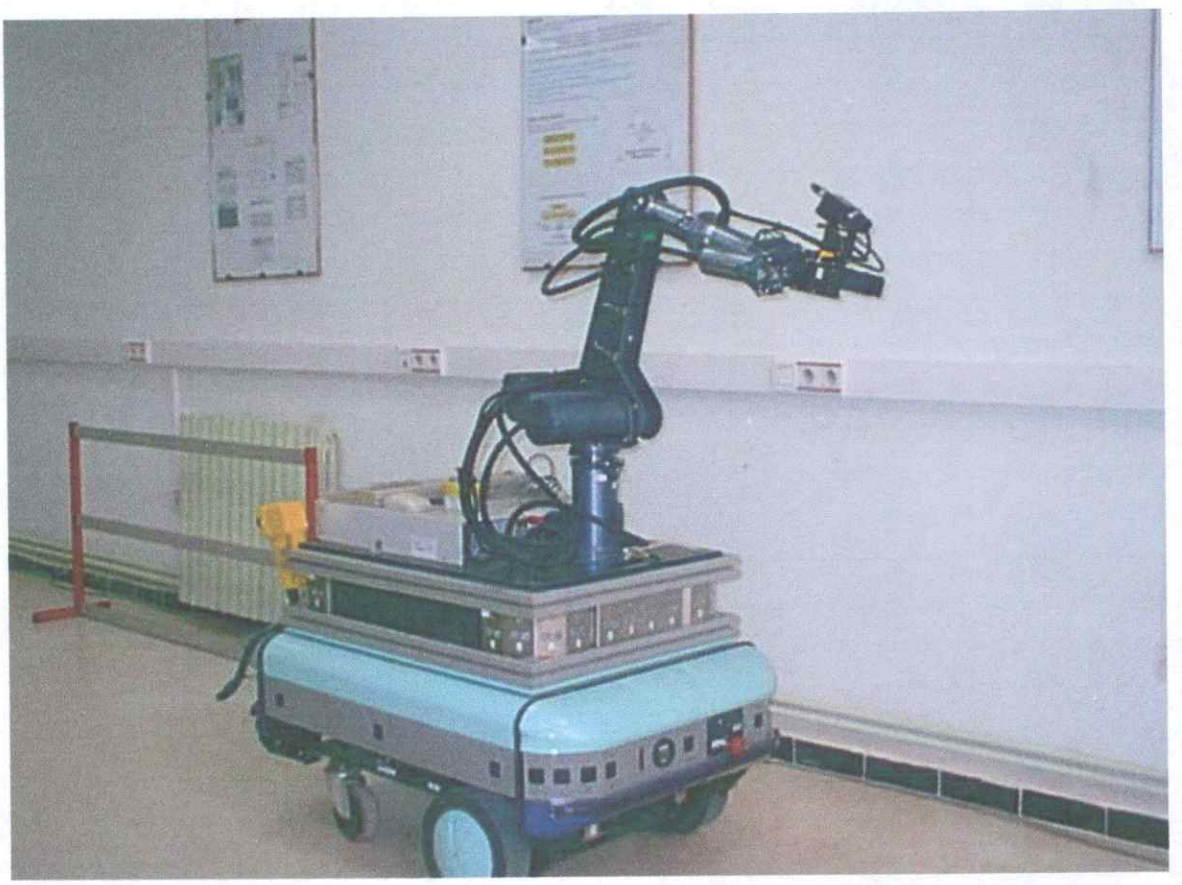


Figure 1 : Robot Robuter-ULM du CDTA

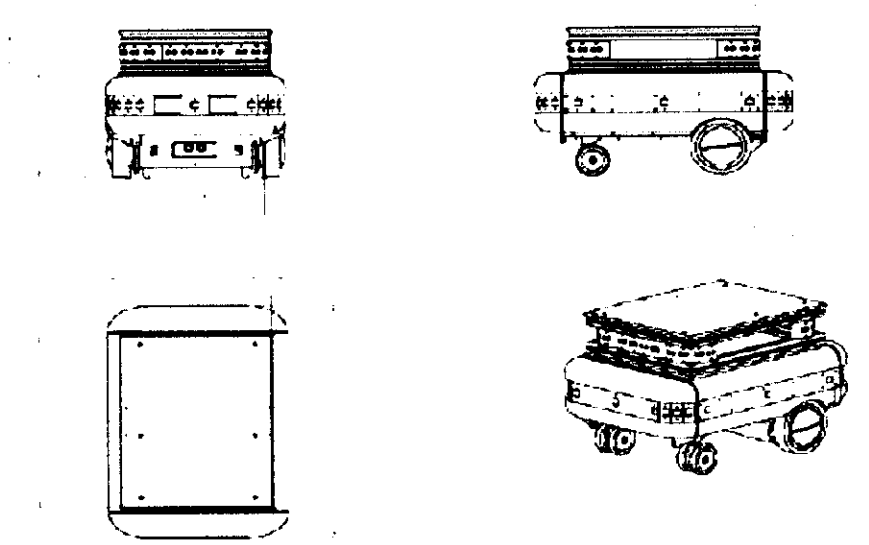
II. Description du Router-ULM

II.1. La base mobile (Robuter)

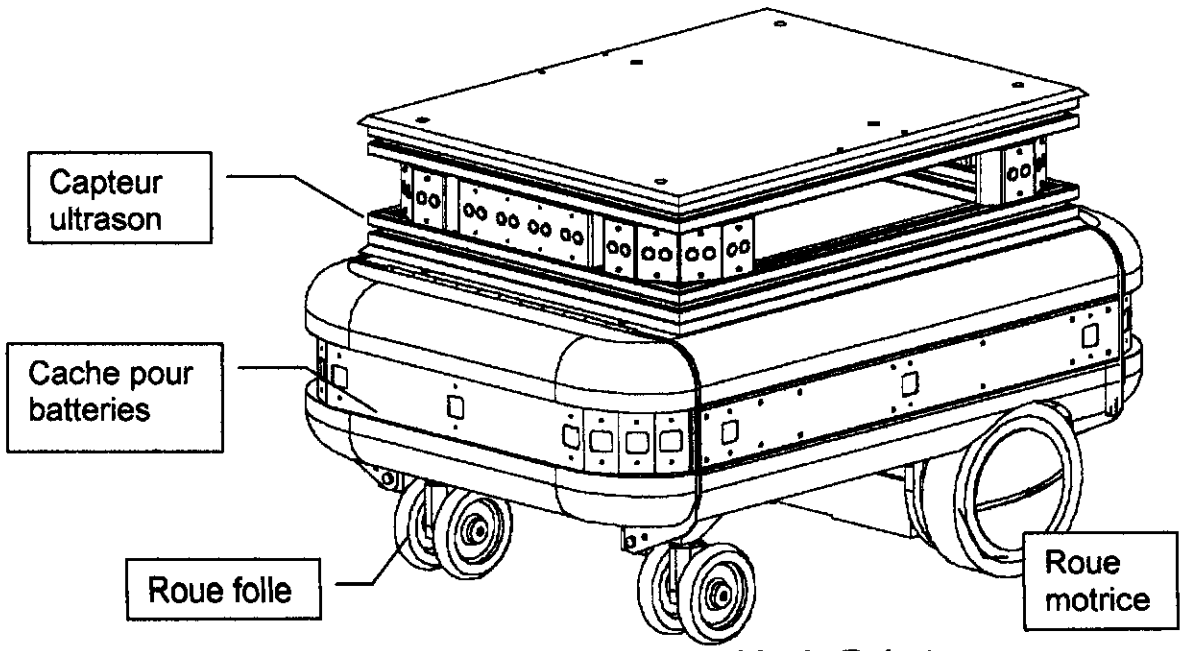
La base mobile comme le montre la Figure 2 est une plate forme à 4 roues, de dimension 695 mm par 1017 mm. Elle pèse environ 150 kg, d'une capacité de charge de 120kg. Deux des 4 roues sont motrices d'une capacité de charge de 15 kg, elles sont actionnées par deux moteurs électriques de 2*300 W à courant continue, sa vitesse varie entre de 5cm/s à 1.25 m/s, avec des roues de 250 mm de diamètre, et un couple nominal de 22 Nm par roue. Les deux autres roues sont des roues folles qui assurent la stabilité de l'ensemble de la plate-forme. La base est équipée de 4

batteries de 12 volts chacune qui fournissent l'énergie nécessaire au fonctionnement du robot.

Le Robuter est une base mobile de type différentielle, cela veut dire que l'orientation de la plate-forme se fait par différences de vitesse des deus roues motrices. La précision de position par encodeurs optiques, et un système d'odomètrie donne la position relative du robot par rapport à un repère lié à la plate-forme.



(a) Vue du Robuter



(b) Vue d'ensemble du Robuter

Figure 2 : Base Mobile (Robuter)

II.2. Les capteurs ultrasons

Le Robuter (base mobile) dispose d'une ceinture de 24 capteurs ultrasons du type SRF04 – Ultrasonic Range Finder, numérotés de 1 à 24. Le premier étant le plus a gauche de la ceinture avant lorsqu'on se met dans le sens du robuter. Ces capteurs servent à détecter les obstacles par envoi et réception d'ondes ultrasonores. Le capteur ultrason SRF04 comme le montre la figure 3 est très compacte de dimensions 43mm x 20mm x 17mm de hauteur. Il arrive à donner des mesures dans un intervalle de 3 cm jusqu'à 3m avec une ouverture angulaire de 30 degrés et une capacité de détecter un objet de 3 cm de diamètre à plus de 2 mètres de distance.

Les 24 capteurs ultrasons sont regroupés en trois groupes par un nombre de huit chacun. Chaque groupe est connecté à une interface digitale d'entrées/sorties à base de microcontrôleurs. Ces cartes sont directement reliées au Pc embarqué via une liaison RS232.

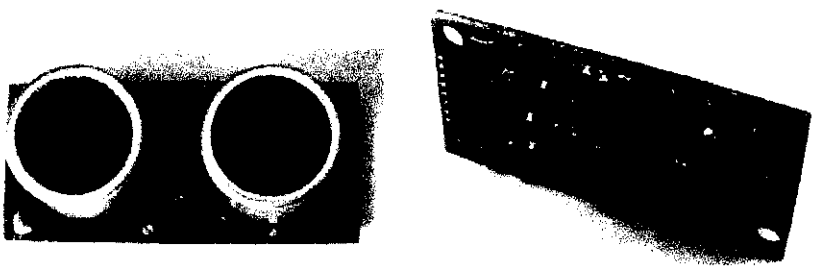


Figure 3 : Capteur Ultrasons

Les 24 capteurs sont disposés sur la plate-forme afin de permettre une totale couverture du robot vis-à-vis de son environnement. Le tableau ci-dessous fourni la position de chaque capteur par rapport au repère lié au robot mobile où X est un axe vers l'avant et qui se trouve dans l'axes du robuter, Y est un axe vers la droite et il se trouve dans l'axe des roues motrices. Par contre, Z est un axe vers le haut du véhicule et Thêta donne l'angle que fait l'axe perpendiculaire à chaque capteur avec l'axe X.

Tableau 1 : Positionnement Des Capteurs Ultrason.

Capteur	X	Z	Y	Theta
1	485.96	428.50	278.50-	90
2	535.98	428.50	-265.10	60
3	572.60	428.50	-228.48	30
4	586.00	428.50	-178.46	0

5	586.00	428.50	-90.00	0
6	586.00	428.50	-30.00	0
7	586.00	428.50	30.00	0
8	586.00	428.50	60.00	0
9	586.00	428.50	178.46	0
10	572.60	428.50	228.48	330
11	535.98	428.50	265.10	300
12	485.96	428.50	278.50	270
13	-39.96	428.50	278.50	270
14	-89.98	428.50	265.10	240
15	-126.60	428.50	228.48	210
16	-140.00	428.50	178.46	180
17	-140.00	428.50	90.00	180
18	-140.00	428.50	30.00	180
19	-140.00	428.50	-30.00	180
20	-140.00	428.50	-90.00	180
21	-140.00	428.50	-178.46	180
22	-126.60	428.50	-228.48	150
23	-89.98	428.50	-265.10	120
24	-39.96	428.50	-278.50	90

II.3. Le bras manipulateur

Le bras manipulateur (ULM) (figure 4) est un bras ultras léger avec 6 axes, muni d'une pince électrique à deux doits à son extrémité. Elle fonctionne en tout ou

rien avec un signal de commande et deux signaux d'état de la pince. Le bras est constitué de segments, de dimensions données dans la figure 5, reliés entre eux par des articulations. La porte du bras est de700 mm avec une répétitivité de +/- 1 mm .Sa capacité de charge est de 2 kg lorsque que le bras est totalement déployé. Le bras est constitue de 7 moteurs DC ,6 pour les 6 axes et un pour la pince. Le mouvement de chaque axe est détecté par un capteur de position (encodeur). Sur chaque axe on trouve une butée électrique et une autre mécanique qui apporte une sécurité au bras contre une mauvaise utilisation au-delà du débattement angulaire permis donnés dans le tableau ci-dessous.

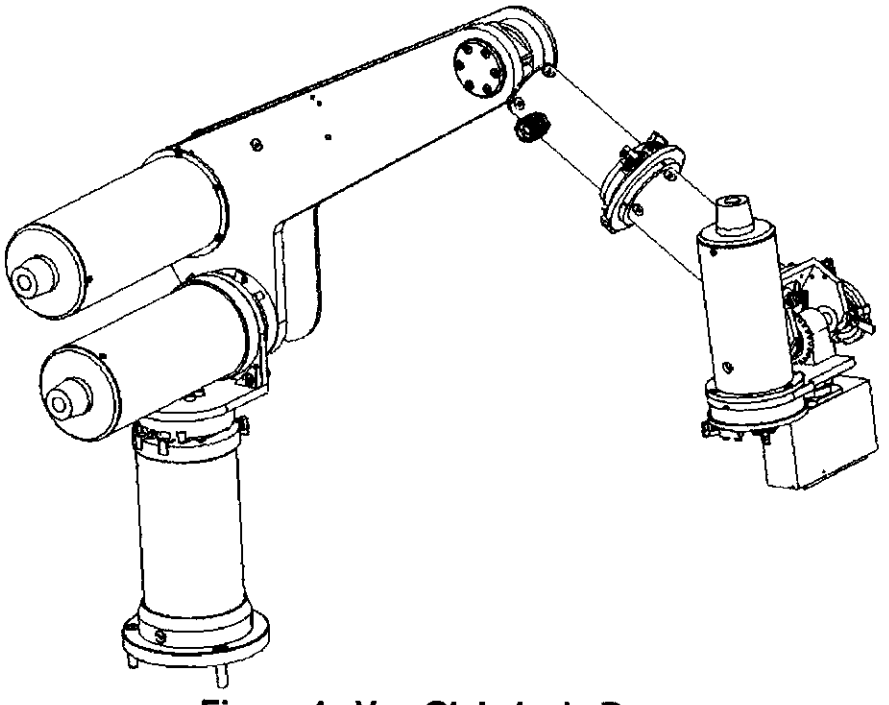


Figure 4 : Vue Globale du Bras

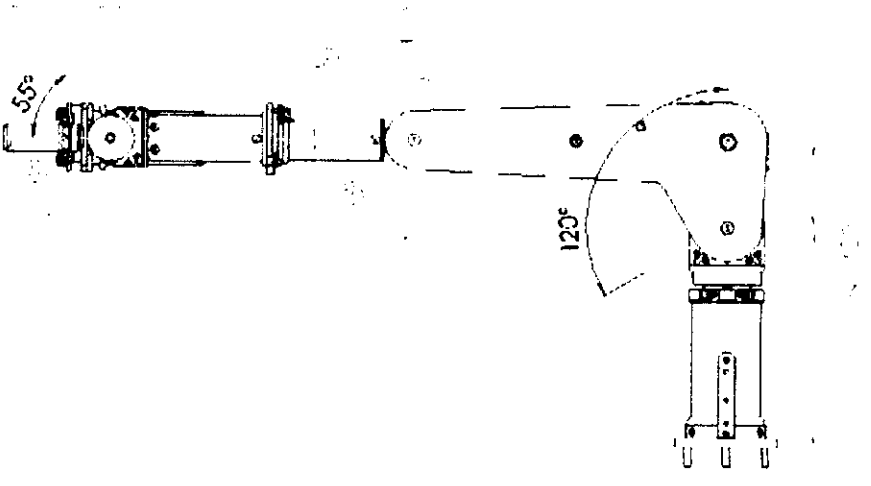


Figure 5: Dimensions du Bras

Tableau 2 : Débattement des Axes

AXE	angle min	angle max
Axe1	-95.4	95.4
Axe2	-22.8	81
Axe3	+10	168
Axe4	-75.5	78
Axe5	-55	55.5
Axe6	-96.8	96.8

II.4. Capteur d'effort

Un capteur d'effort est placé sur l'organe terminal du robot manipulateur avant la pince. Le capteur d'effort a pour fonction de fournir les forces et les couples exercés par le robot sur son environnement. L'importance d'un tel dispositif est de permettre l'implémentation d'une commande en force. Ce genre de commande est fortement présent dans certaines applications en robotique qui demande une interaction entre le robot et son environnement.

Le capteur placé sur le bras UL est de type Gamma SI-32-2.5 de ATI Industrial Automation. C'est un capteur 6 axes qui permet la mesure les trois forces selon les axes X, Y et Z ainsi que les trois couple autour de ces trois axes. Le fonctionnement de ce capteur est basé sur le principe piézoélectrique qui permet de transformer à l'aide d'un transducteur toute force ou couple en un signal électrique.

II.5. Caméra

Une caméra de type XC-ST50CE (figure 6) de chez Sony placée sur l'organe terminal du bras. Cette caméra est compacte avec une taille de 44 (l) x 29 (h) x 57,5 (p) mm et légères (seulement 110 g). C'est caméra noir et blanc offrent des images d'excellente qualité et une très grande sensibilité. Elle est dotée de la dernière génération de capteurs d'images CCD Sony. Elle est idéale pour les applications exigeantes de l'industrie, de la microscopie, du traitement d'images et de la vision industrielle.

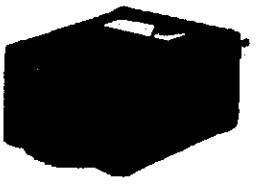


Figure 6 : Caméra XC-ST50CE

La caméra est connectée à une carte d'acquisition de type PCI Frame Grabber Modèle 611. La carte d'acquisition placée sur le bus PCI. Elle permet la capture des images monochrome qui proviennent de la caméra et stocker dans la mémoire RAM du Pc embarqué.

II.6. LAN sans fil

Afin de garantir une liberté de déplacement, une liaison sans fil entre le robot et le point d'accès au réseau est nécessaire. Pour réaliser cette liaison on a utilisé un produit de l'entreprise BREEZECOM : le BreezeNET PRO.11. Il se compose de deux produits qui réalisent une transmission transparente point à point entre le point d'accès au réseau (par le BreezeNET PRO.11 AP-10) et le robot (par le BreezeNET PRO.11 SA-10). La configuration est visible sur la figure 7.

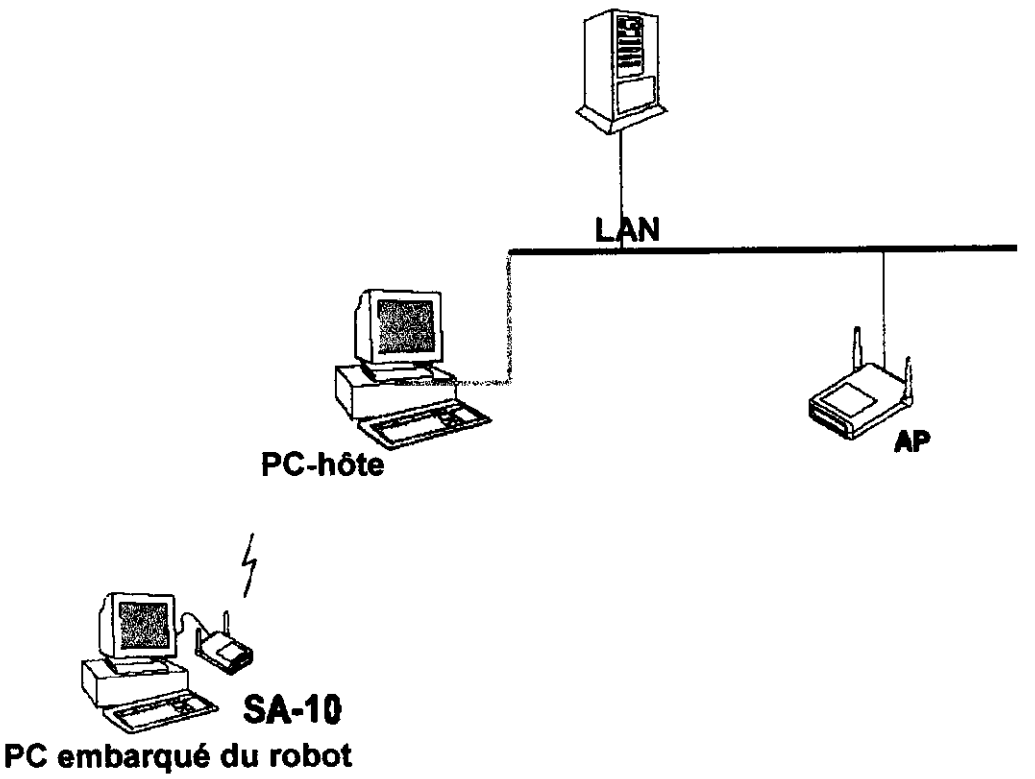


Figure 7: Système LAN sans Fil

III. Architecture matérielle du robot

Le robot manipulateur mobile possède une architecture matérielle (figure 8) distribuée comprenant quatre cartes Robosoft à base de microcontrôleur MPC555 de Motorola et d'un Pc embarqué. Les deux moteurs de la base mobile sont contrôlés par une carte MPC555, deux autres cartes permettent le contrôle des six axes du bras UL et de la pince. La dernière carte permet l'acquisition des mesures données par le capteur d'effort. Tout système est supervisé par un Pc embarqué de type industriel Intel MMX 233 qui permet à l'utilisateur l'accès aux différentes ressources du robot.

Les quatre cartes ainsi que le Pc embarqué sont reliés par un bus CAN qui permet l'échange du flux de données entres les différents processeurs. Mise à part les

capteurs ultrasons et la caméra toutes les données des ressources du robot transitent sur le bus CAN. Les capteurs ultrasons sont connectés directement au Pc embarqué via une liaison RS232. Quant à la caméra, un câble coaxiale la relié à la carte d'acquisition.

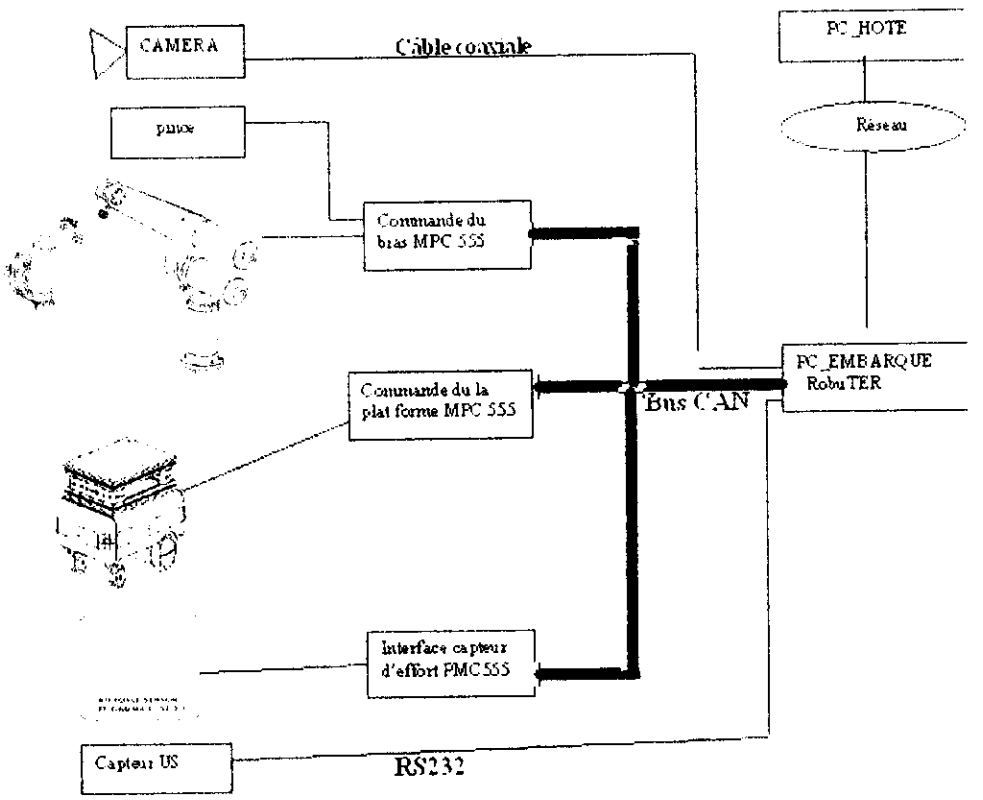


Figure 8 : Architecture matérielle du Robuter-ULM

III.1. Carte Robosoft MPC555

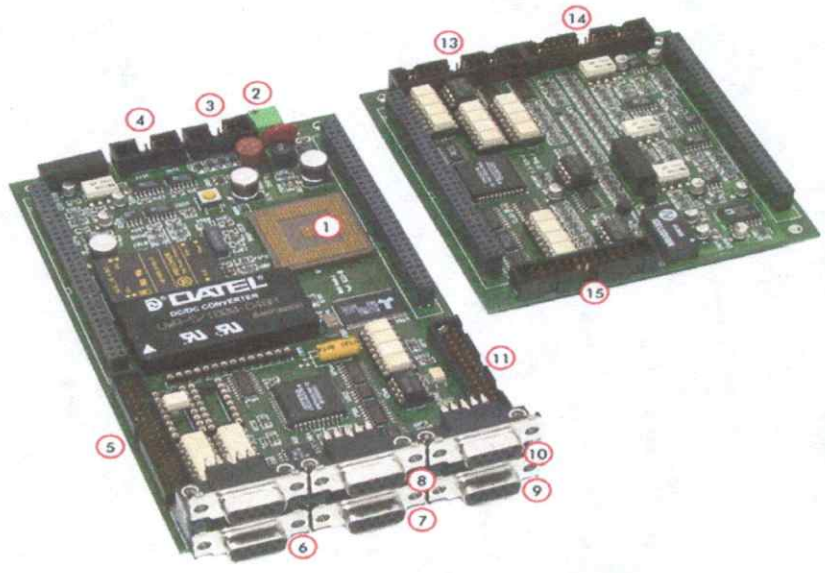


Figure 9 : Carte Robosoft MP C555

La carte Robosoft MPC555 est conçue pour la commande de robot, de véhicule électrique ou toute autre machine. Elle permet de commander un axe grâce à l'ajout d'une carte fille le contrôle s'étend jusqu'à 4 axes. Cette carte est basée sur le microcontrôleur MPC555 32-bit de Motorola. Elle peut opérer à une fréquence de 40 MHz. Divers signaux logique et analogique peuvent être utilisé pour l'acquisition de et la commande de matériel. Les sorties disponibles peuvent être analogique ou PWM. On y trouve :

(1) Connecteur du chip MPC555
(2) Alimentation continue
(3) Interface BDM (Basic Debug Interface)
(4) Entré Analogique (Joystick,...)
(5) Entré/Sortie logiques
(6) Ligne série Synchrone (Codeur absolu)
(7) Ligne série Asynchrone (Port 0)
8) Ligne série Asynchrone (Port 1)
(9) Connecteur du bus CAN (Port 0)
(10) Connecteur du bus CAN (Port 1)
(11) Axe 1
(13) Axe 2
(14) Axe 3
(15) Axe 4

IV. Architecture logiciel [4]

Afin d'exploiter les ressources du robot, l'application utilisateur doit contenir deux parties : une partie haut niveau en C/C++ et une autre bas niveau sous syndex avec

un échange de données entre les deux partie en utilisant la couche RTAI de linux comme le montre la figure 10.

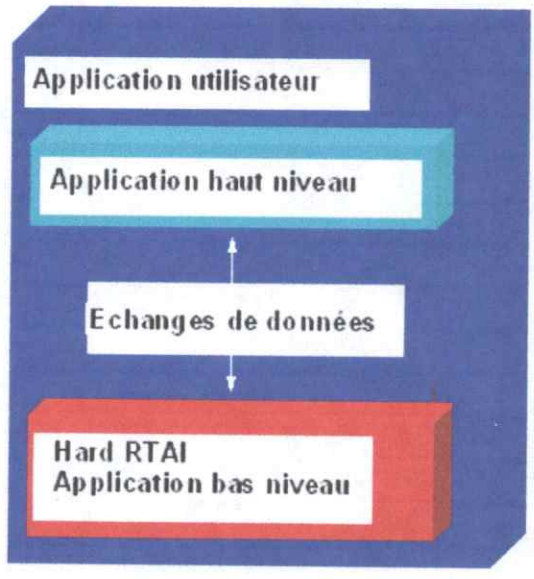


Figure 10 : Architecture Logicielle

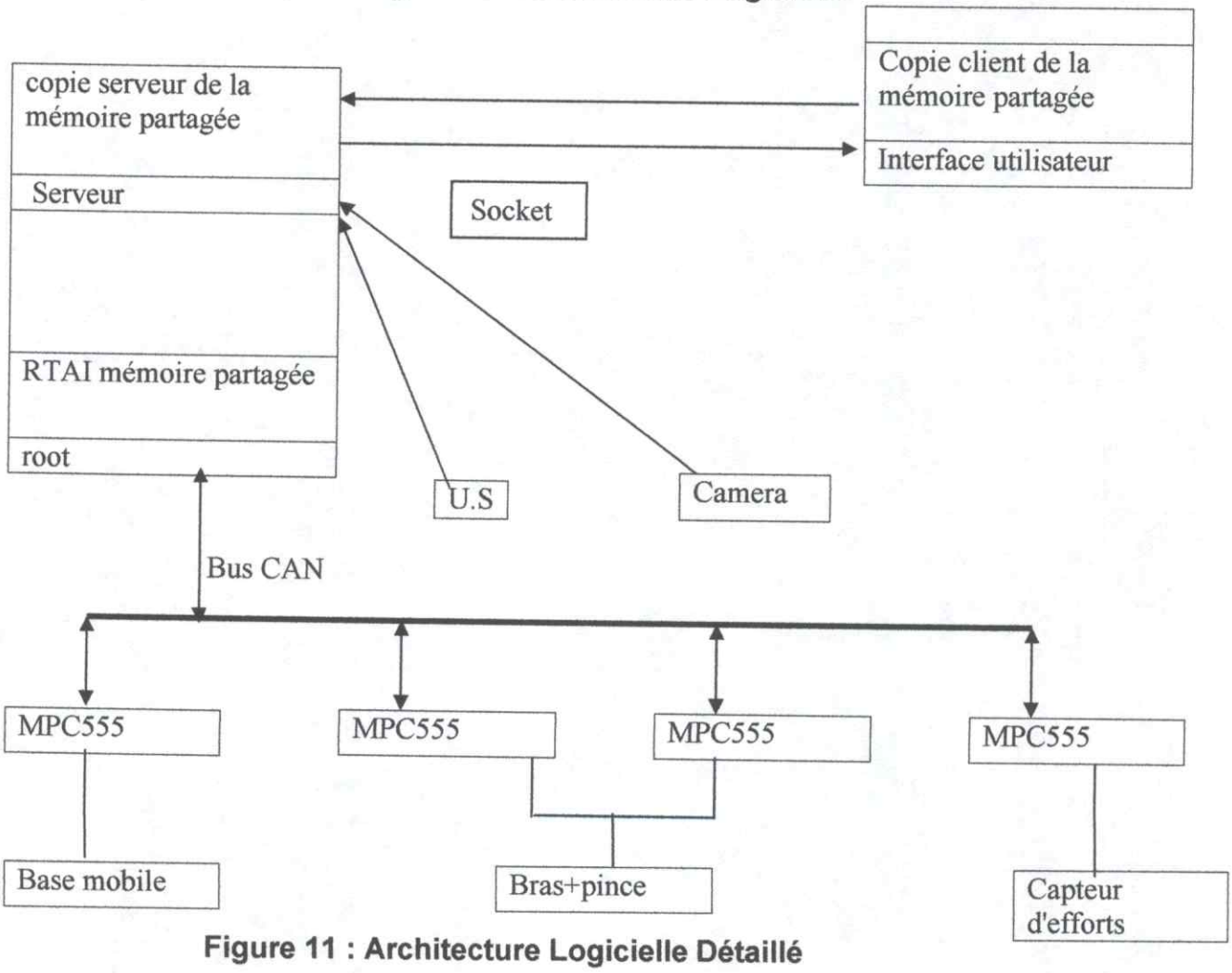


Figure 11 : Architecture Logicielle Détaillé

On entend par une application haut niveau tout programme comme la planification de trajectoires, stratégies de navigation, etc. …
L'échange de données se fait par le biais des mémoires partagées qui font parti du noyau RTAI et de l'espace utilisateurs de Linux.
Toute ressource présente sur la carte Robosoft peut être partagée sans aune restriction.

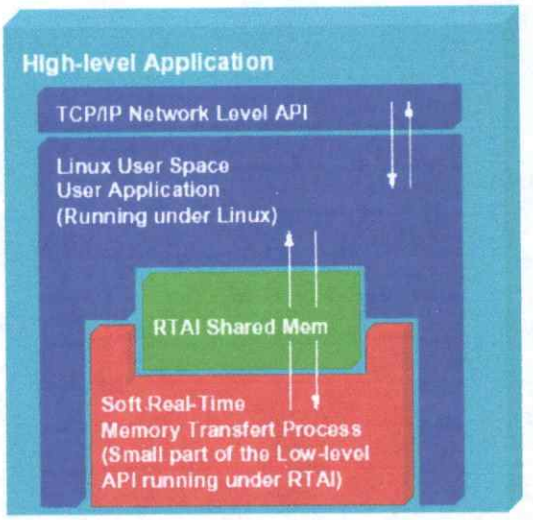


Figure 12 : Application Haut Niveau

Une application bas niveau est un programme conçu sous syndex comme la commande des moteurs, la lecture des capteurs, etc. …
Les ressources matérielles sont contrôlées par plusieurs cartes MPC555. Ces cartes s'échangent des données via le bus CAN. Par conséquent les entrées/sorties sont prises en charge par la source logicielle dédiée qui s'exécute dans les cartes MPC555 puis la couche RTAI est utilisée pour la mise à jour de mémoires partagées pour l'espace utilisateurs linux d'échange de données.

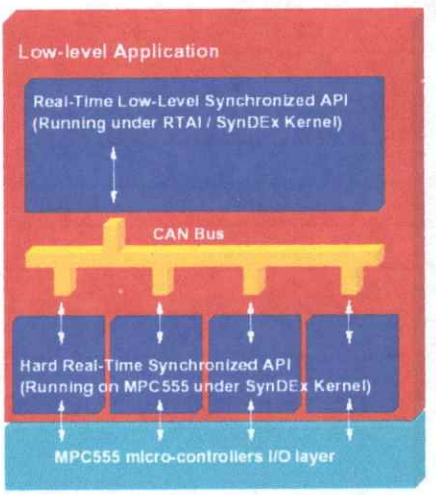


Figure 13 : Application Bas Niveau

IV.1 Méthodologie AAA

C'est une méthodologie fonde sur des modèles de graphes, autant pour spécifier les algorithmes applicatifs et les architectures matérielles destribuées, que pour déduire des implantations possibles en termes de transformation de graphes.

IV.2 Syndex

Syndex est un logiciel de CAO (Conception Assisté par Ordinateur) niveau système, fondé sur la méthodologie adéquation algorithme-architecture (AAA) pour le prototypage rapide et l'optimisation d'applications distribuées temps réel embarquées complexes. Il permet de spécifier à l'aide de graphes les algorithmes applicatifs et les architectures distribuées, de lancer des heuristiques d'optimisation de l'implantation des algorithmes sur les architectures conduisant à une simulation temporelle de l'exécution temps réel, et enfin de générer automatiquement des exécutifs distribués temps réel sans interbloquage et à faible surcoût.

CHAPITRE III

CONTROLE A DISTANCE

I.Introduction

Le contrôle à distance selon l'explication anglaise du terme (remote control) est le contrôle et la commande d'un robot ou d'une machine quelconque à distance. C'est une technologie en plein épanouissement que ce soit sur le plan local (réseaux locaux) ou pour des réseaux non déterministe comme internet.
Le principe consiste à déplacer les commandes distante, pour que l'opérateur puisse avoir les informations à distance (images, localisation, retours d'efforts,....), cette distance varie de quelques mètres a plusieurs kilomètres, la liaison peut être de tout type (câble, fibre optique, micro-ondes).

II. Le concept de contrôle à distance [5,2]

Dans notre cadre, le robot manipulateur mobile a pour but de fournir à un agent humain un outil potentiellement mobile qui lui permette d'intervenir à distance sur un milieu particulier afin de remplir un objectif donné. Cette définition permet d'associer à cette branche de la robotique de nombreuse d'applications. De ces applications nous pouvons citer l'exploration dans des milieux inconnus, la surveillance industrielle, et l'intervention dans des milieux hostiles.

L'utilisation de robots manipulateur mobile à distance découle du besoin à un moment précis d'agir sur un environnement sans pouvoir (ou vouloir) être présent physiquement. Ils découlent donc tous de la volonté de fournir à un opérateur une sorte d'extension de son propre corps lui permettant d'agir à distance. La façon dont est réalisée cette commande à distance va elle aussi être amenée à varier. La distance séparant le robot de l'opérateur, le médium de communication utilisable, la quantité d'information échangeable ou encore la complexité du milieu dans lequel intervient le robot vont amener à répartir différemment les rôles à assumer par l'opérateur et par le robot. Entre deux usages d'un robot d'intervention, la succession d'actions entreprises, le site d'intervention, les difficultés rencontrées vont changer. Le robot et l'opérateur doivent s'adapter aux conditions de chaque opération à exécuter.

Dans le domaine de la commande à distance nous pouvons distinguer différents types qui peuvent être classifiés selon deux critères :

Premièrement suivant la nature de liaison entre le robot et l'opérateur :

- Liaisons mécaniques.

Cette première méthode et sans doute la plus « ancienne » consiste à transmettre les ordres de l'opérateur au robot au moyen d'une liaison mécanique permettant de reproduire le mouvement souhaité. Cependant, cette mécanique de liaison limite fortement la distance à laquelle l'opérateur peut se trouver.

- Transmission électrique filaire :

  Dans ce genre de commande à distance, on supprime la liaison mécanique pour la remplacer par un médium véhiculant ordres, consignes ou tensions pour la commande du robot. On peut dès lors augmenter considérablement la distance entre l'opérateur et le robot.

- Transmission sans fil :

  Ce mode regroupe l'ensemble des transmissions non filaires disponibles. Même si la portée varie en fonction du moyen de communication utilisé (radio fréquences, infra-rouges ou ondes acoustiques), ce mode de transmission permet de s'affranchir de l'entrave que représente le lien filaire. Le robot n'est donc plus lié physiquement au poste de contrôle. Cette solution est une alternative au lien filaire comme par exemple avec les engins d'exploration spatiaux, lorsque l'on souhaite une grande mobilité du robot (domaine accessible plus important, moins de contraintes sur la trajectoire). Cela soulève toutefois des problèmes de débit d'information, de portée, de brouillage et de perte de contrôle du robot lorsque la communication est masquée ou perdue.

- Commande via un mode de transmission mixte filaire et non filaire :

  Ce dernier type de commande s'est considérablement développé avec l'apparition du réseau international qu'est Internet. L'idée de pouvoir commander un robot à partir de n'importe quel point de la « toile » ouvre des perspectives intéressantes en termes de ressources partagées et de souplesse d'utilisation. Le principe consiste à connecter l'opérateur au robot en passant d'abord par le réseau Internet puis par une borne émettrice (radio par exemple) communiquant avec le robot. On allie ainsi les avantages d'une liaison non filaire de courte ou moyenne portée (peu onéreuse et transportable) avec une grande distance entre l'opérateur et le robot. Cependant, le fait de passer par un réseau dont l'accès est non déterministe et dont la circulation de l'information peut varier dans l'espace (en fonction de l'itinéraire choisi par les routeurs) et dans le temps (en fonction de l'encombrement du réseau) présente des difficultés supplémentaires pour la commande à distance. En effet, un réseau comme Internet génère des retards variables et potentiellement des ruptures de communication, qui sont incompatibles avec une commande à distance "classique" consistant à faire parvenir au robot les commandes qu'il doit appliquer à ses actionneurs. Les retards dans le sens opérateur-robot entraînent une action décalée dans le temps, source d'instabilité d'autant plus sensible si le contrôle sur le robot est effectué en vitesse ou en effort. Dans l'autre sens, c'est la réaction de l'opérateur qui va être liée avec une situation qui n'est pas celle que rencontre le robot à l'instant présent. De même, si le robot doit faire face à l'apparition soudaine d'un obstacle sur sa trajectoire (un trou détecté au dernier instant par exemple), il faut être capable de réagir en un temps fini qui est incompatible avec les retards variables générés par Internet.

Dans nos travaux nous avons considéré ce mode de commande à distance. Malheureusement le réseau du CDTA souffrant d'une certaine saturation nous provoque des retards considérables. Pour palier à ce problème nous avons opté pour un réseau local entre le robot et l'opérateur distant.

Une deuxième classification de la commande à distance est possible en considérant le mode utilisé. Ce mode correspond au type d'interaction qui existe entre l'opérateur et le robot. Selon le mode d'interaction, le rôle de l'opérateur au sein de la commande et le niveau d'autonomie du robot sont différents. Les figures 1 à 3 illustrent l'ensemble des modes de commande à distance en présentant ces derniers suivant un ordre croissant quant à l'autonomie opérationnelle et décisionnelle dont est doté le contrôleur embarqué du robot.

- Robot non autonome :

  La figure 1a) présente le type de commande à distance de plus bas niveau. L'ensemble du contrôle du robot se trouve déporté du coté de l'opérateur. Le robot est alors défini comme non autonome puisqu'il est entièrement dépendant de la communication qui fait partie intégrante de la boucle d'asservissement. L'impact des retards de communication est alors directement répercuté sur la stabilité de la commande du robot. Dans la commande à distance de type consigne (figure 1b)), le robot acquiert un peu d'autonomie opérationnelle en embarquant les asservissements qui lui permettent de générer la commande.

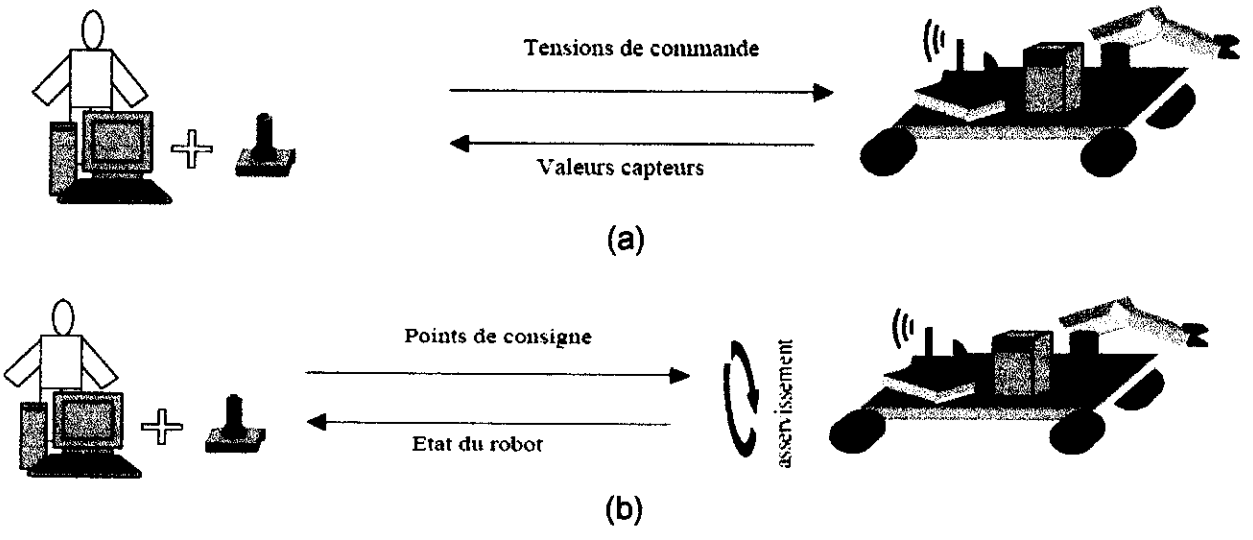


Figure 1. Commande à distance bas niveau

Des retards dans la transmission des consignes au robot n'occasionnent pas forcement une instabilité du robot. Cependant, le robot ne dispose tout de même que d'une autonomie plus que limitée. Là encore, le robot ne possédant aucune autonomie décisionnelle, il ne réalise aucune supervision.

- Robot à autonomie opérationnelle

La figure 2 présente une commande à distance de type tâche. Le robot dispose en plus des asservissements, de la possibilité de générer lui-même ses trajectoires. Il gagne encore en autonomie opérationnelle puisqu'il assume de plus en plus d'aspects dans la construction de sa commande. L'opérateur dispose donc de la totalité des décisions à prendre ce qui soumet toujours le robot aux aléas de communication.

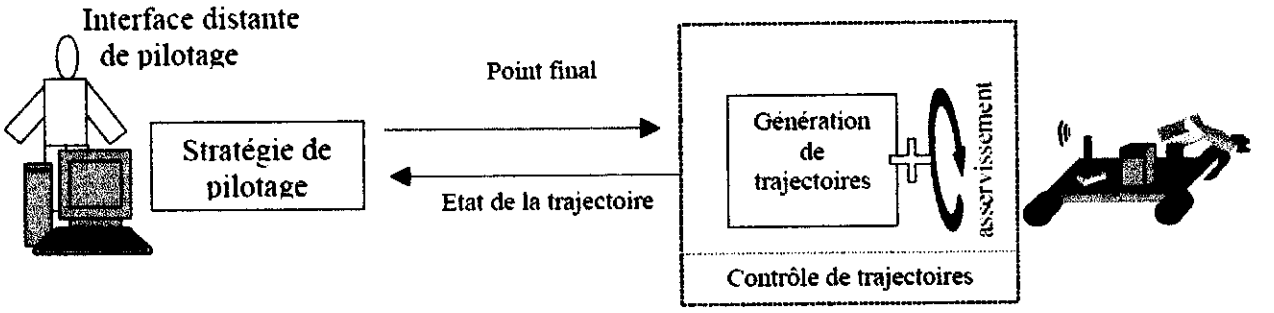


Figure 2 : Commande à distance à autonomie opérationnelle

- Robots à autonomie décisionnelle :

Les figures 3 a) et b) présentent les deux derniers types de commande à distance avant le robot entièrement autonome où l'opérateur n'intervient plus dans la boucle de contrôle. Le premier mode dit de type objectif permet de donner au robot des directives quant à ce qu'il doit accomplir. Le contrôleur embarqué « évolué » que nous appelons superviseur d'objectif permet de gérer et coordonner l'ensemble des ressources du robot afin de remplir un objectif donné. Cette séquence d'opérations peut être adaptée en fonction de paramètres liés à la situation du robot et à la perception qu'il a de son environnement. Il dispose dès lors d'un pouvoir décisionnel lui permettant de choisir sa séquence d'actions. L'opérateur n'intervient que pour transmettre les différents objectifs au robot, ou en tant que recours dans les situations où le robot ne trouve pas de solution. L'opérateur est indispensable dans ce schéma où l'intelligence embarquée reste limitée.

Le dernier type de commande à distance accroît encore le pouvoir décisionnel du robot. En effet, l'opérateur se contente dans ce dernier cas de fournir au robot une mission de façon très agrégée telle que "trouver un objet" ou "effectuer un prélèvement à tel endroit". En plus de son superviseur d'objectif, le robot dispose d'un planificateur qui va construire une séquence de sous-objectifs qui vont lui permettre d'accomplir sa mission. Ici, l'impact de la communication est minime et le robot peut parfaitement se trouver coupé de

l'opérateur pendant un certain temps sans que l'accomplissement de sa mission s'en ressente.

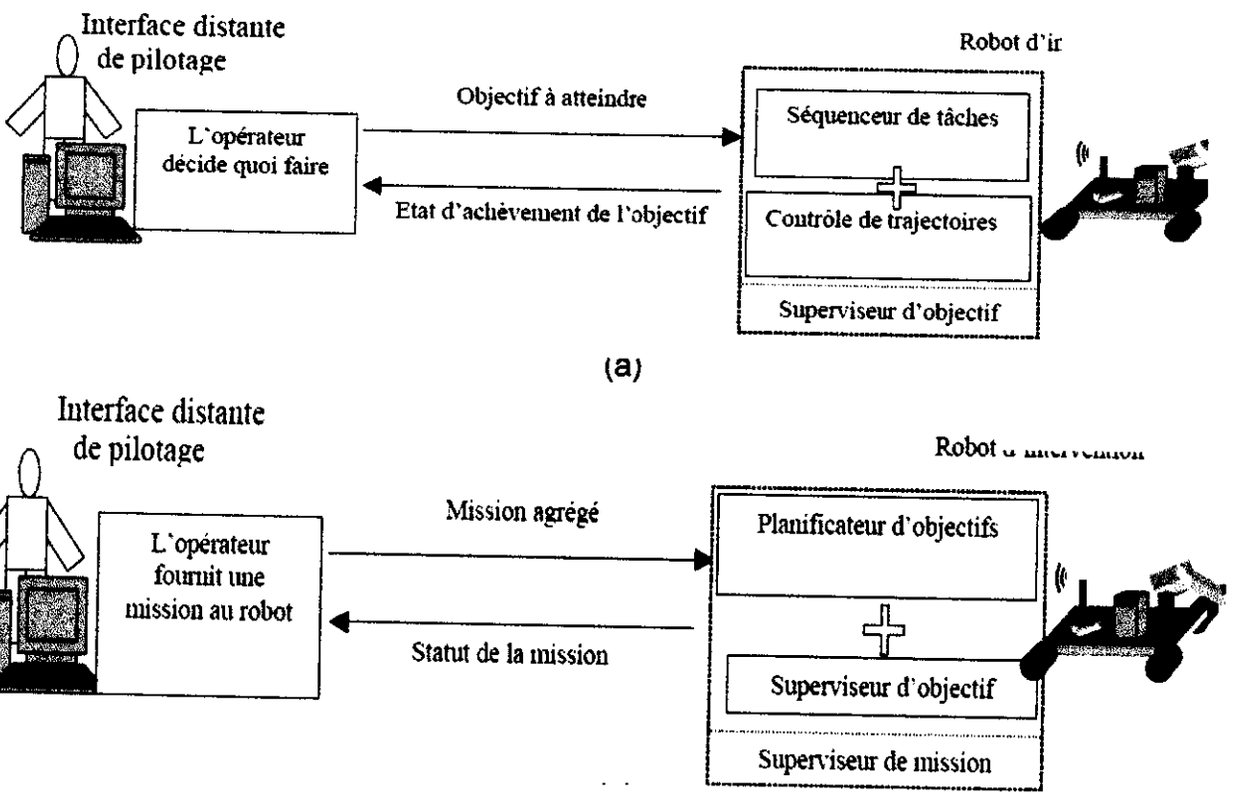


Figure 3 : Commande à distance à autonomie décisionnelle

Comme le robot du CDTA a été fourni totalement ouvert et vierge, nous nous consacrerons ici à une commande à distance de type bas niveau. Ce type de commande à distance n'exclue pas toutefois la possibilité de réaliser des commandes à autonomie opérationnelle. Cela peut constituer les travaux futurs sur ce site. Rien n'empêche, en effet, la coexistence de plusieurs modes avec passage (commutation) automatique de mode de commande à distance directe vers un mode autonome dès lors qu'un événement ne permettant plus la commande à distance directe survient (retard important). Le robot assume alors seul sa mise en situation stable et sûre. Cette commutation automatique de mode de commande à distance n'est toutefois pas traitée dans ce document.

Dans la suit du paragraphe nous présentons les outils informatiques qui nous permettent l'implémentation d'une commande à distance du robot robuter-ULM.

III Introduction aux réseaux [6,7]

Un réseau est un ensemble d'éléments ou d'objets interconnectes entres eux, qui permettent la circulation d'information entre eux. On peu parler de plusieurs réseaux: De réseaux téléphoniques, d'un réseau hydraulique pour les canalisations d'eaux, d'un réseau informatique,...etc.

Pour nous ce qui nous intéresse c'est les réseaux informatiques, un réseau informatique est un ensemble d'ordinateurs relies entre eux par des liens physiques (câble coaxial, fibre optique...) ou par des liaisons sans fil, et qui permettent l'échange de données entre eux (sous forme d'impulsions électrique, de lumière, ou d'ondes électro-magnetique pour les réseaux sans fil).

Les réseaux qui permettaient à leurs origine de relier des terminaux passifs aux ordinateurs centraux, permettent actuellement de relier tout type d'ordinateur (gros serveurs, stations de travail, terminaux ...).il sont donc indispensables pour les entreprises et les administrations (gestion, commerce, base de données, recherche), ainsi que pour les particuliers (Internet, jeux en réseau).

On distingue plusieurs types de réseaux:

- Réseau local (LAN) : qui peut s'étendre de quelques mètres a quelques kilomètres et qui satisfait les besoins internes des entreprises.

- Réseau métropolitain (MAN) : qui relie plusieurs sites situés dans la même ville, chacun possédant son propre réseau local, par exemple les différents sites d'une administration.

- Réseau étendu (WAN) : qui permet de communiquer a l'échelle d'un pays ou du monde entier, comme Internet, les infrastructure pouvant être terrestres ou spatiales à l'aide des satellites de télécommunications.

On distingue aussi plusieurs topologies de réseau : les réseaux en étoile, en anneau, en bus simple, en boucles et tout autre forme.

Il existe deux modes de réseaux : le mode point à point et le mode diffusion.

- Pour le mode point à point, le support relie une paire d'équipement seulement, les éléments non directement connectés le font par l'intermédiaire des autres noeuds du réseau.

- Pour le mode diffusion, tous les élément partagent le même support de transmission.Chaque donnée envoyée par un élément est reçue par tous les autres éléments, le destinataire reconnaîtra le message lui est destine grâce a l'adresse qu'il véhicule, un seul élément peut envoyer une donnée a un moment donné, c'est pour ça qu'il faut que l'émetteur écoute au préalable si la voie est libre. Dans une telle configuration, la rupture du support entraîne l'arrêt du

réseau, par contre l'arrêt d'une machine n'influe pas les autres, ce mode est très utilise dans les réseaux locaux.

On ne peut parler de réseaux informatiques sans parler de son plus grand problème qui est l'interconnexion des réseaux différents , au début des années 70, chaque constructeur apportait sa propre solution réseau (SNA d'IBM, DSA de Bull, TCP/IP..), il était impossible d'interconnecter ces réseaux si une norme n'était pas faite , et c'est pour ça que l'ISO (International Standard Organisation) a établie la norme OSI (Open Système Interconnections).Cela permet a tout système ouvert que se soit un ordinateur, un terminal ou un réseau respectant cette norme d'échanger des informations avec des équipements différents issus d'autres marques respectant cette norme aussi.

L'un des rôles essentiels de la norme était de définir un modèle pour toute les architectures, basé sur un découpage en sept couches, chacune ayant une fonction particulière.

7 Application
6 Présentation
5 Session
4 Transport
3 Réseau
2 Liaison
1 Physique

Figure 4 : Les sept couches de référence du modèle OSI de l'ISO

Chaque couche est constituée d'éléments matériels et logiciels et offre un service à la couche située immédiatement au-dessus d'elle en lui épargnant les détails de l'implémentation nécessaires. Chaque couche "n" gère la communication avec la même couche "n" de la machine distante en suivant un protocole bien spécifique.

- La couche physique : elle fournit les moyens mécaniques, électriques, fonctionnels et procéduraux nécessaires a l'activation, au maintien et à la désactivation des connexions physiques destinés à la transmission de bits entre deux entités de liaison de données.

- La couche liaison : elle fournit les moyens fonctionnels et procéduraux nécessaires à l'établissement, au maintien et à la libération des connexions de liaison de données entre entités du réseau. Elle détecte et corrige, si possible, les erreurs dues au support physique et signale au réseau les erreurs irrécupérables. Elle supervise le fonctionnement de la transmission et définit la structure syntaxique des messages, la manière d'enchaîner les échanges selon un protocole normalisé ou non.

- La couche réseau: elle assure toutes les fonctionnalités de relaie et d'amélioration de services entre entités de réseau, à savoir: l'adressage, le routage, le contrôle de flux et la détection et correction d'erreurs non réglées par la couche liaison.

- La couche transport: elle assure un transfert transparent entre entités de session, en les déchargeant des détails d'exécution. Elle a pour rôle d'optimiser l'utilisation des services du réseau disponible afin d'assurer au moindre coût les performances requises par la couche session.

- La couche session: elle fournit aux entités de la couche présentation les moyens d'organier et de synchroniser les dialogues et les échanges de données.

- La couche présentation: elle s'occupe de la syntaxe et de le sémantique des informations transportés, en se chargeant notamment de la représentation des données.

- La couche application: la couche application donne au processus d'application le moyen d'accéder à l'environnement OSI et fournit les services directement utilisables par l'application, qui sont le transfert de données, l'allocation de ressource, l'intégrité et la cohérence des données accédés et la synchronisation des application coopérantes.

IV. Introduction aux protocoles [8]

Un protocole est une méthode standard permettant la communication entre deux machines, c'est-à-dire un ensemble de règles et de procédures à respecter pour émettre et recevoir des données sur un réseau. Il en existe plusieurs selon nos attentes de la communication. Certains protocoles sont par exemple, spécialisés dans l'échange de fichiers (le FTP), d'autres peuvent servir à gérer simplement l'état de la transmission et des erreurs (c'est le cas du protocole ICMP),...

Sur Internet, les protocoles utilisés font partie d'une suite de protocoles, c'est-à-dire un ensemble de protocoles reliés entre-eux. Cette suite de protocoles s'appelle TCP/IP.
Elle contient, entre autres, les protocoles suivants:

-HTTP ;
-FTP ;
-ICMP ;
-TCP ;
-UDP ;
-SMTP ;
-NNTP ;
-Telnet ...

On classe généralement les protocoles en deux catégories selon le niveau de contrôle des données que l'on désire:

- Les protocoles orientés connexion: Il s'agit des protocoles opérant un contrôle de transmission des données pendant l'établissement d'une communication machines. Dans un tel schéma, la machine réceptrice envoie des accusés de réception lors de la communication, ainsi la machine émettrice est garantie de la validité des données qu'elle envoie. Les données sont ainsi envoyées sous forme de flot. TCP est un protocole orienté connexion.

- Les protocoles non orientés connexion: Il s'agit d'un mode de communication dans lequel la machine émettrice envoie des données sans prévenir la machine réceptrice, et la machine réceptrice reçoit les données sans envoyer d'accusé de réception à la première. Les données sont ainsi envoyées sous forme de blocs (datagrammes). UDP est un protocole non orienté connexion.

V. Le protocole TCP/IP [7, 8,9]

V.1. Introduction

TCP/IP est née de la réflexion de chercheurs américains suite à un problème posé par l'armée américaine , celle-ci disposait de plusieurs bases sur le territoire, chacune disposant de sa propre logistique informatique, donc de différentes machines reliées entre elle par des réseaux locaux, étant donnée que ces bases était reliées entre elle par des câbles, le problème était de trouver un moyen pour que l'information puisse circuler en reconfigurant le système automatiquement en cas de ruptures de liaison pour retrouver le chemin adéquat. De là, est né le protocole IP (Internet Protocole ou Interconnected Network Protocol).

IP est un protocole de la couche 3 du modèle OSI (la couche réseau), il assure sans connexion non fiable la délivrance de datagramme IP, et qui a pour fonction essentielle le routage, la définition du format des datagrammes ip qui est l'unité de base des données circulant sur le réseau, et la définition de la gestion de la remise non fiable des datagrammes.

Le problème de ce protocole est qu'il envoi l'information d'une machine à l'autre alors que l'information s'échange d'une application a l'autre, et c'est pour ça qu'on a crée le protocole TCP (Transport Control Protocol).

TCP est l'un des principaux protocoles de la couche transport (couche 6) du modèle OSI, il gère les données provenant ou à destination de la couche inférieure (le protocole IP) au niveau des applications. C'est un protocole orienté connexion qui permet à deux machine de communiquer en contrôlant l'état de la transmission, il assure un service fiable. Grâce à ce protocole la couche Internet ne se préoccupe que de l'envoie de données sous forme de datagramme sans se préoccuper du contrôle de données qui est assuré par le protocole TCP.

Donc le nom TCP/IP a été choisi en référence à ces deux principaux protocoles qui le caractérisent. Aujourd'hui TCP/IP intègre beaucoup d'autre protocole (ICMP, IGP, FTP, SMTP, HTTP, ...).

TCP/IP est très répandu à cause de robustesse prouvée (quelques millions de machines interconnectées dans le monde). Il est également très répandu, car dès son origine, il a été implémenté sur des systèmes Unix. Beaucoup de chercheurs ayant contribué à l'évolution de TCP/IP à son origine sont issus de l'université de Berkeley qui a très largement diffusé son système Unix avec l'interface des sockets pour manipuler des connexions TCP/IP.

V.2. Le protocole TCP

TCP est un protocole de la couche 6 (application) du modèle OSI, qui permet la gestion des données au niveau application, c'est un protocole orienté connexion, c'est-à-dire que l'état de la transmission est contrôlé. Les applications dialoguant lors d'une connexion sont considérées l'une comme un serveur et l'autre comme un client, qui doivent établir la connexion avant de pouvoir dialoguer. Après l'établissement de la connexion, les deux machines s'échangent des messages spécifiques. Cette connexion est bidirectionnelle simultanée (full duplex) composée de deux flots de données indépendants et de sens contraires. Les données sont encapsulées c'est à dire qu'on ajoute aux paquets de données un entête qui va permettre la synchronisation de la transmission et d'assurer leurs réception.

TCP échange entre les deux machines un flux d'octet non interprété par TCP, c'est aux applications des deux extrémités que revient ce travail.
Dans le cas d'informations trop volumineuses, elles sont fractionnées en données de taille optimale par TCP. De même TCP peut regrouper des données pour former qu'un seul datagramme de taille convenable pour décharger le réseau, cette unité est appelée segment. La figure () montre le format des données TCP.

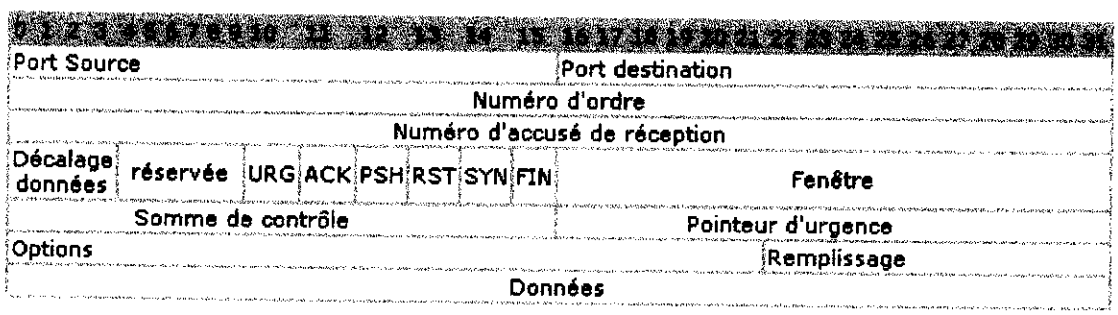


Figure5 : Format des données TCP

Tableau 1 : Explication des différents champs d'une données TCP

Port Source	16 bits	Port relatif à l'application en cours sur la machine source.
Port Destination	16 bits	Port relatif à l'application en cours sur la machine de destination.
Numéro d'ordre	16 bits	Lorsque le drapeau SYN est à 0, le numéro d'ordre est celui du premier mot du segment en cours Lorsque SYN est à 1, le numéro de séquence est le numéro de séquence initial utilisé pour synchroniser les numéros de séquence (ISN).
Numéro d'accusé de réception	32 bits	Dernier segment reçu par le récepteur.
Décalage des données	4 bits	Il permet de repérer le début des données dans le paquet Le décalage est ici essentiel car le champ d'options est de taille variable.
Réservé	6 bits	Champ inutilisé actuellement mais prévu pour l'avenir.
Drapeaux (oflags)	6x1 bits	Les drapeaux représentent des informations supplémentaires: URG: si ce drapeau est à 1 le paquet doit être traité de façon urgente ACK: si ce drapeau est à 1 le paquet est un accusé de réception PSH (PUSH): si ce drapeau est à 1, le paquet fonctionne suivant la méthode PUSH RST: si ce drapeau est à 1, la connexion est réinitialisée SYN: si ce drapeau est à 1, les numéros d'ordre sont synchronisés FIN: si ce drapeau est à 1 la connexion s'interrompt
Fenêtre	16 bits	Champ permettant de connaître le nombre d'octets que le récepteur souhaite recevoir sans accusé de réception.
Somme de contrôle	Checksum ou CRC	La somme de contrôle est réalisée en faisant la somme des champs de données de l'en-tête, afin de pouvoir vérifier l'intégrité de l'en-tête.
Pointeur d'urgence	16 bits	Indique le numéro d'ordre à partir duquel l'information devient urgente.
Options	Taille variable	Des options diverses.

Remplissage		On remplit l'espace restant après les options avec des zéros pour avoir une longueur de 32 bits.

Les principales caractéristiques du protocole TCP sont:

- Permet de remettre en ordre les datagrammes en provenance du protocole IP ;
- Permet de vérifier le flot de données afin d'éviter une saturation du réseau ;
- Permet l'initialisation et la fin d'une communication de manière courtoise.
- Permet de formater les données en segments de longueur variable afin de les remettre au protocole IP.

Il a aussi deux très importantes caractéristiques qui sont:

-La fonction de multiplexage :

C'est faire transiter sur une même ligne des données provenant d'applications diverses. Ces opérations sont réalisées grâce au concept de ports (ou sockets), c'est-à-dire un numéro associé à un type d'application, qui, combiné à une adresse IP, permet de déterminer de façon unique une application qui tourne sur une machine donnée.

-Fiabilité des transferts :

TCP est un protocole fiable. Il possède un système d'accusé de réception qui assure la bonne réception des données. Lors de l'émission d'un segment, un numéro de séquence lui est associé. A la réception de ce segment, la machine réceptrice va renvoyer un accusé de réception (un segment de donnée dont le drapeau ACK est à 1) munis d'un numéro égal au numéro de la séquence précédant.

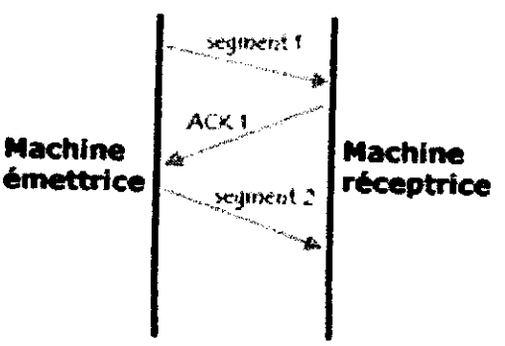


Figure 6 : Fiabilité des transferts

Aussi à chaque envoi d'un segment, la machine émettrice enclenche une minuterie, qui si elle expire et que l'accusé de réception n'arrive pas, alors dans ce cas la machine émettrice considère que le segment est perdu et l'envoi de nouveaux.

Toutefois, si le segment perdu arrive à destination en retard, alors la machine réceptrice saura que c'est un doublon grâce à son numéro de séquence, et l'éliminera.

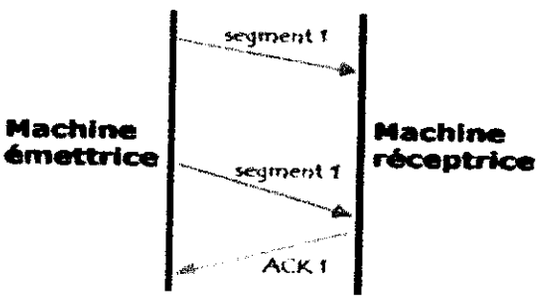


Figure 7 : Fiabilité des transferts

V.3 La connexion TCP

Etablissement d'une connexion :

Etant donné que ce processus de communication, qui se fait grâce à une émission de données et d'un accusé de réception, est basé sur un numéro d'ordre (appelé généralement numéro de séquence), il faut que les machines émettrices et réceptrices (client et serveur) connaissent le numéro de séquence initial de l'autre machine.

L'établissement de la connexion entre deux applications se fait souvent selon le schéma suivant:

- Les ports TCP doivent être ouverts ;
- L'application sur le serveur est passive, c'est-à-dire que l'application est à l'écoute, en attente d'une connexion ;
- L'application sur le client envoie une requête de connexion au serveur dont l'application est en ouverture passive. L'application du client est dite « en ouverture passive ».

- Les deux machines doivent donc synchroniser leurs séquences grâce à un mécanisme commun appelé three ways handshake (poignée de main en trois temps), que l'on retrouve aussi lors de la clôture de session.

Ce dialogue permet d'initier la communication, il se déroule en trois temps, comme sa dénomination l'indique:

- Dans un premier temps, la machine émettrice (le client) transmet un segment dont le drapeau SYN est à 1 (pour signaler qu'il s'agit d'un segment de synchronisation), avec un numéro d'ordre N, que l'on appelle numéro d'ordre initial du client.

- Dans un second temps, la machine réceptrice (le serveur) reçoit le segment initial provenant du client, puis lui envoie un accusé de réception, c'est-à-dire un segment dont le drapeau ACK est à 1 et le drapeau SYN est à 1 (car il s'agit là encore d'une synchronisation). Ce segment contient le numéro d'ordre de cette machine (du serveur) qui est le numéro d'ordre initial du client. Le champ le plus important de ce segment est le champ accusé de réception qui contient le numéro d'ordre initial du client, incrémenté de 1.

- Enfin, le client transmet au serveur un accusé de réception, c'est-à-dire un segment dont le drapeau ACK est à 1, et le drapeau SYN est à zéro (il ne s'agit plus d'un segment de synchronisation). Son numéro d'ordre est incrémenté et le numéro d'accusé de réception représente le numéro de séquence initial du serveur incrémenté de 1.

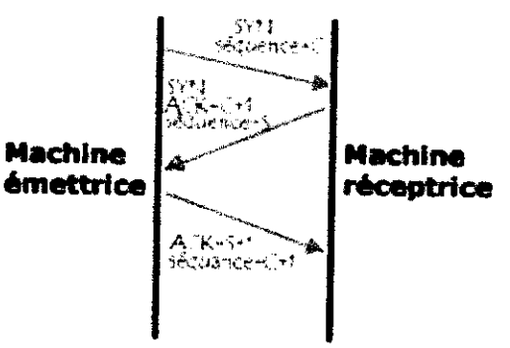


Figure 8 : Synchronisation de deux machines.

Suite à cette séquence comportant trois échanges, les deux machines sont synchronisées et la communication peut commencer.

Fin d'une connexion :

Le client peut demander à mettre fin à une connexion au même titre que le serveur.
La fin de la connexion se fait de la manière suivante:

- Une des machines envoie un segment avec le drapeau FIN à 1, et l'application se met en état d'attente de fin, c'est-à-dire qu'elle finit de recevoir le segment en cours et ignore les suivants ;
- Après réception de ce segment, l'autre machine envoie un accusé de réception avec le drapeau FIN à 1 et continue d'expédier les segments en cours.
- Suite à cela, la machine informe l'application qu'un segment FIN a été reçu, puis envoie un segment FIN à l'autre machine, ce qui clôture la connexion....

V.4. Adressage

Chaque ordinateur d'un réseau Internet possède une adresse IP unique codée sur 32 bits. Une adresse est représenté de 4 nombres séparés par des points dans une "notation décimale pointe" avec un octet pour chaque nombre, chaque nombre est compris entre 0 et 255. Une adresse IP est constitue d'une paire (id. de réseau, id. de machine) et appartient a une certaine classe (A, B, C, D et E) selon la valeur de son premier octet.

Donc on a pour le premier octet de :

- 0 a 127 : classe A
- 128 a 191 : classe B
- 192 a 223 : clase C
- 224 a 239 : classe D
- 240 a 247 : classe E

Les adresses de classe A sont utilisées pour les très grands réseaux qui comporte plus de 65536 ordinateur comme celui du pentagone ou du MIT, le nombre de réseaux de classe A est limite a 127 dans le monde.

Les adresses de classe B sont réservées aux réseaux ayant entre 256 et 65536 ordinateurs.

Pour la classe C on ne peut dépasser les 256 machines, le nombre de réseaux de ce type peut dépasser 2 millions.

Certaines adresses IP ont une signification particulières on peut citer :

- 0.0.0.0 utilise par une machine pour connaître sa propre adresse IP.
- <id.de réseau>.<id.de machine nul> permet de désigne le réseau lui-même.
- <id.de réseau>.<id.de machine avec tous ses bits a 1> est une adresse de diffusion (broadcasting)
- les adresses de classe a de 10.0.0.0 a 10.255.255.255, de classe b de 172.16.0.0 à 176.31.255.255 et de classe c de 192.168.0.0 a 192.168.255.255 sont réservées à la constitution de réseaux intranet.

Le système des adresse IP permet aussi la création de sous-reseaux en découpant la partie réservée à l'adresse machine sur un réseau en deux parties dont la première sera un identificateur de sous-reseaux.

VI. Le modèle client/serveur [6,10]

Le modèle client/serveur est apparu dans les année 90 de l'aboutissement d'un ensemble d' évolutions technologiques (capacité mémoire, performance des processeurs et des réseaux, évolution des logiciel : interface graphique, multimédia, interface de communication) , pour que tout utilisateur d'une entreprise ou autre puisse accéder à tout information autorisée par les règles de confidentialité et de sécurité, de

manière instantanée depuis n'importe quel poste en utilisant une interface aussi simple que possible.

Une architecture client/serveur signifie que des machines clientes (des machines faisant partie du réseau) contactent un serveur, une machine généralement très puissante, qui leur fournit des services. Ces services sont des programmes fournissant des données telles que l'heure, des fichiers, une connexion, ...

Cette architecture comporte plusieurs caractéristiques, on peut citer les plus importantes:

- Le serveur est fournisseur de service et le client est consommateur.
- C'est toujours le client qui déclenche la demande de service. Le serveur attend passivement la requête du client.
- Un serveur traite plusieurs clients au même temps et contrôle leur accès aux ressources.
- Il est possible d'ajouter et de retirer des stations clientes. Il est possible de faire évoluer les serveurs.
- Le modèle client-serveur est indépendant des plates-formes matérielles et logicielles.
- On peut modifier le serveur sans toucher au client. La réciproque est vraie.

Cette architecture est utilisée dans plusieurs domaines telle: la gestion de base de données, systèmes transactionnels, système de messagerie, web, Internet....etc, vu tous les avantages qu'elle comporte (des ressources centralisées: étant donné que le serveur gère des ressources communes à tous les utilisateurs, une meilleure sécurité: car le nombre de points d'entrée permettant l'accès aux données est moins important, une administration au niveau serveur , un réseau évolutif ou l'on peut supprimer ou ajouter des clients sans perturber le réseau).

Néanmoins, cette architecture a quelque inconvénients telle que le coût élevé du à la technicité du serveur, et aussi ,le maillon faible que constitue le serveur étant donné que toutes l'architecture tourne autour de lui donc dépendante de lui.

La figure montre le fonctionnement d'une architecture client/serveur

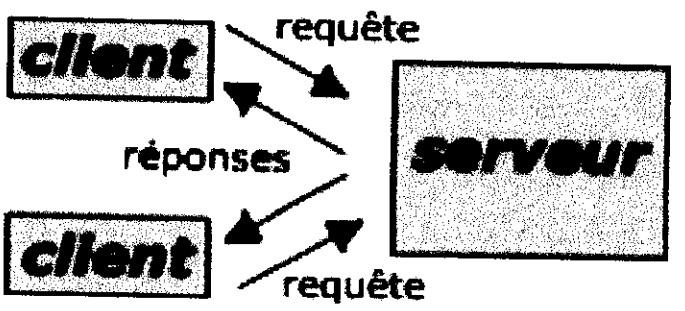


Figure 9 : Fonctionnement d'une architecture client/serveur

VI.1 les socket

On va parler maintenant d'un mécanisme important pour l'implémentation d'une architecture client/serveur qui est le socket.

Dans une connexion client/serveur chaque extrémité utilise une socket, on a une socket client et une autre socket serveur.

La socket est identifié grace à son adresse IP et à son numéro de port.

La socket client demande une connexion à une socket serveur.

La socket serveur attend une demande de connexion puis dialogue avec le ou les sockets clientes dont la requête est acceptée.

Quand un processus serveur gère chaque requête de façon indépendante, il est qualifié de serveur itératif. Il gère les requêtes une par une dans leurs ordre d'arrivée mais il n'y a pas de chevauchement entre les traitements de requêtes.

A l'inverse, quand on ne peut pas prévoir le temps nécessaire à un serveur pour répondre, on dit qu'il s'agit d'un serveur concurrent. Ce serveur crée un processus distinct pour chaque requête de service. Typiquement, à chaque fois qu'une connexion est établie, le serveur lance un nouveau processus qui reste en écoute d'éventuelles autres demandes de connexion. Cela nécessite un environnement multi-tâches.

Dans le cas du protocole TCP, ou l'on travaille en mode connecté, la figure montre une socket en mode connecte.

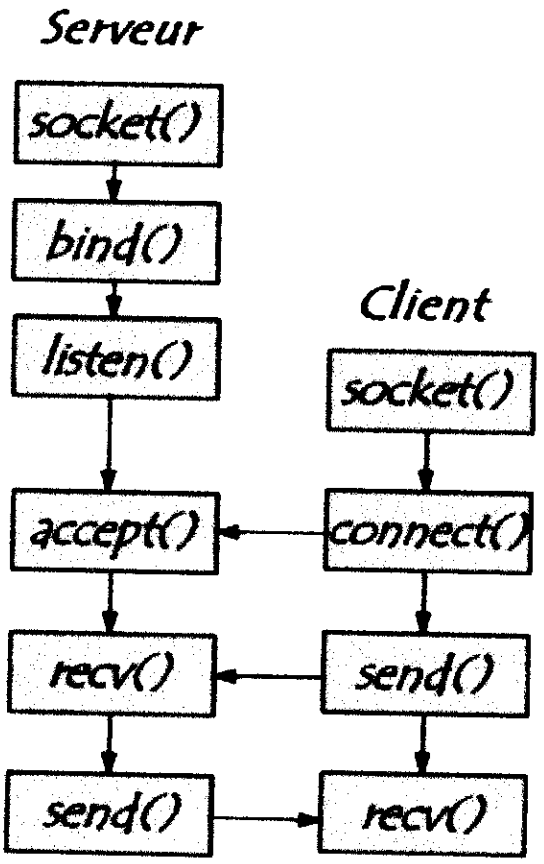


Figure 10 : Socket en mode connecté

Dan le cas du protocole UDP, ou l'on travail en mode deconnecté, la figure () montre une socket en mode deconnecté.

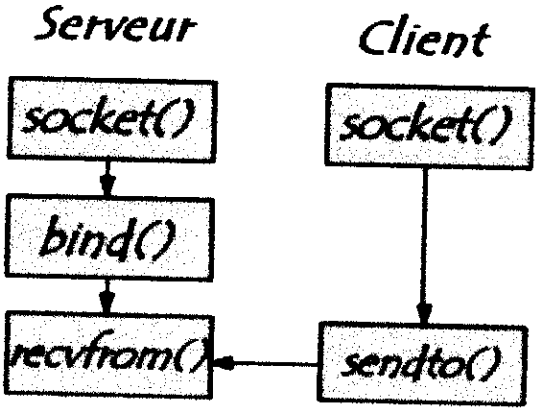


Figure 11 : Socket en mode deconnecté.

Voici les définitions des fonctions citées là-dessus qu'on verra plus en détail dans le chapitre 5:

Socket(): pour la création de la socket d'écoute.

Bind(): pour l'attribution du port d'écoute.

Listen(): pour l'écoute du port, et l'attente d'une éventuelle demande de connexion de la part d'un client.

Accep() : pour l'acceptation du client et la creation d'un socket de communication.

Recv() : pour la réception de données.

Send() : pour l'émission de données.

Connect() : pour la tentative de connexion au serveur de la part du client.

CHAPITRE IV

CONCEPTION DE L'APPLICATION LOGICIELLE

I. Méthodologie de l'application [11,12]

I.1 Historique d'UML

L'approche objet est depuis longtemps une réalité, les concepts de base sont stables est largement approuvés. L'approche objet a commencé à apparaître vers la fin des années 60 , avec Simula qui a été le premier langage de programmation à implémenter le concept de type abstrait a l'aide des classes, et en 1976 Smaltalk implémente les concepts fondateurs de la méthode objet : encapsulation , agrégation et héritage . Les premiers compilateurs C++ font leur apparution en début des années 80 et de nombreux langages oriente objet voient le jour (Eiffel, loops...).

Actuellement l'approche objet est devenue incontournable, dés lors qu'on cherche à concevoir des logiciels complexes qui doivent résister a des évolutions incessantes, la programmation objet bénéficie d'une panoplie d'outils et de langages performants.

Mais l'approche comporte quelques inconvénients, qui peuvent induirent leurs utilisateurs dans l'erreur et faire couler le projet.

De ces difficultés on peut citer :

- L'approche objet n'est pas très intuitive, car il est difficile pour le cerveau humain de décomposer un problème informatique en termes d'objet et d'interaction entre ces objets, et rien dans les concepts de base de l'approche objet ne dicte comment modéliser la structure objet d'un système de manières pertinentes.
- L'application des concepts objet nécessite beaucoup de rigueurs, car il y a beaucoup d'embiguités et d'incompréhensions, et cela est due au fait que beaucoup de développeur pensent souvent objet qu'a travers les langages de programmation, alors que ceux-ci ne valident en rien l'utilisation de moyens techniques pour concevoir un système conforme a la philosophie objet.

C'est pour cela qu'il il faut disposer d'un outil qui sera un guide dans l'utilisation des concepts objets.

Et pour cela il nous fau t:

1) un langage (pour s'exprimer clairement à l'aide des concepts objets), qui doit permettre de

- représenter des concepts abstraits (graphiquement par exemple),
- limiter les ambiguïtés (parler un langage commun, au vocabulaire précis, indépendant des langages de programmation orientés objet),
- faciliter l'analyse (simplifier la comparaison et l'évaluation de solutions).

2) une démarche d'analyse et de conception objet pour :

- Ne pas effectuer une analyse fonctionnelle et se contenter d'une implémentation objet, mais penser objet dès le départ.
- Définir les vues qui permettent de décrire tous les aspects d'un système avec des concepts objets.

C'est pour ça que les langages de modélisation orientés objets ont fait leur apparition, au milieu des années 70, quand les spécialistes ont commencé à expérimenter de nouvelles approches d'analyse et de conception. Entre 1989 et 1994 le nombre de méthodes objet est passé de 10 a 50, sans qu'elles répondent vraiment aux besoins des utilisateurs.

Basées sur l'expérience acquise, de nouvelles méthodes sont apparues, les plus importantes sont Booch, OOSE (Object Oriented Software Engineering) et OMT (Object Modeling Technique), elles ont été reconnues au niveau mondial comme étant incontournable. Chacune de ses méthodes constituait une méthodes complète mais présenter encore des inconvénients, en résumé chacune des méthodes avait un créneau dans lequel elle excellait.

Au milieu des années 90 les principaux auteur des méthodes, Booch, OOSE et OMT que sont respectivement Grady Booch (Rational Software Corporation), Ivar Jacobson (objectory) et James Rumbaugh (General Electric) ont commencé à adopter les idées des deux autres, puis se sont décidés à travailler ensemble pour la création d'un langage de modélisation unifié, et cela pour trois raisons :

- Poursuivre cette évolution ensemble pour éliminer les différences inutiles qui aurait embrouillé les idées des utilisateurs.
- Apporter une stabilité au marche orienté objet.
- Apporter des améliorations aux trois méthodes et répondre a des problèmes qu'aucune d'elle ne traiter de manière satisfaisante.

Et pour atteindre es objectifs suivants :

1-La modélisation des systèmes au moyens de techniques orienté objet, depuis leur conception jusqu'à leur artefact exécutable.

2-La résolution des problèmes d'échelles inhérentes aux systèmes complexes et essentiels.

3-La création d'un langage de modélisation utilisable à la fois par les humains que les machines.

De là, donc est née l'UML (Unified Modeling Language) traduit (langage de modélisation objet unifié) de la fusion des trois méthodes (Booch, OMT, OOSE), qui est le fruit d'un large consensus, de très nombreux acteurs industriels de renoms (HP,

IBM, ORACLE, MICROSOFT.....) l'ont adopte et participe à son développement .en un peu de temps UML est devenus un standard incontournable.

I.2 Description d'UML

UML (Unified Modeling Language) est un langage standard conçu pour l'écriture des plans d'élaboration de logiciel. Il peut être utilisé pour visualiser, spécifier, construire et documenter les artefacts d'un système à fortes composantes logicielles.

UML est utilisé pour visualiser dans le sens où il permet d'utiliser au mieux les représentations graphiques ou textuelles en fonction des besoins, où chaque symbole UML possède une sémantique bien définie, et chaque développeur ou outil peut interpréter ce modèle sans ambiguïté.

UML est un langage pour spécifier dans le sens où il construit des modèles précis, sans ambiguïté et complets. Et cela en spécifiant toutes les décisions importantes en termes d'analyse, de conception, et d'implémentation.

UML est un langage pour construire dans le sens où l'on peut construire ou traduire les modèles d'UML dans un langage de programmation tel que JAVA, C++..... Et cela s'appelle l'application de l'igenierie vers l'aval. L'inverse est aussi possible c'est-à-dire passer du code vers le modèle ou plus precesimment la reconstruction du modèle à partir du code. Cela s'appelle la retro-ingenierie, mais il nécessite le support d'un outil pour cause de perte d'informations.

UML est un langage pour la documentation dans le sens où il permet de documenter l'architecture d'un système dans ses moindres détails.

UML est utilisé dans beaucoup de domaine tel :

- Les services informatiques d'entreprise.
- Les services bancaires et financiers.
- Les télécommunications.
- Les transports.
- La défense et l'aero-spatiale.
- Le commerce de détail.
- L'électronique de détail.
- Les sciences.
- Les services distribués basé sur le web mais aussi à des systèmes qui n'appartiennent pas à cette catégorie, comme le workflow d'un système judiciaire, ou la conception de matériel informatique.

Les points forts d'UML:

UML est un langage formel et normalisé : gain de précision, gage de stabilité, encourage l'utilisation d'outils.

UML est un support de communication performant :

- Il cadre l'analyse.
- Il facilite la compréhension de représentations abstraites complexes.
- Son caractère polyvalent et sa souplesse en font un langage universel.

Les points faibles d'UML

- La mise en pratique d'UML nécessite un apprentissage et passe par une période d'adaptation.
- Le processus qui est non couvert par UML est une autre clé de la réussite d'un projet, Or, l'intégration d'UML dans un processus n'est pas triviale et améliorer un processus est une tâche complexe et longue.

Composition d'UML

Il se compose de trois éléments essentielles: les briques de base, les règles qui déterminent la manière d'assembler les briques de base, quelques mécanisme généraux.

a-les briques de base

Il existe trois sortes de brique

-les éléments

- les relations

-les diagrammes (on les verras lors de l'établissions des différents diagrammes).

-les éléments:

-les éléments structurels :

C'est les parties les plus statiques du modèle, il représente des éléments conceptuels ou physiques et il en existe sept.

-La classe: elle représente un ensemble d'éléments qui partagent les mêmes attributs, les même opérations, les mêmes relation et la même sémantique, elle implémente une ou plusieurs interfaces.

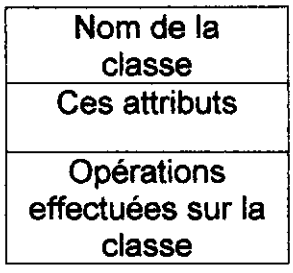


Figure 1 : Classe

-Interface: c'est un ensemble d'opérations qui définissent la fonction d'une classe ou d'un composant, et qui décrit le comportement apparent de cet élément.

Figure 2 : Interface

-Collaboration : elle définit une interaction et constitue une société de rôle et de divers éléments qui travaillent ensemble pour fournir un comportement coopératif qui présente une utilité supérieure à la somme de toutes les parties.

Figure 3 : Collaboration

-Cas d'utilisation : c'est la description d'une séquence d'actions exécutées par un système, pour produire un résultat qui peut être constaté par un acteur particulier.

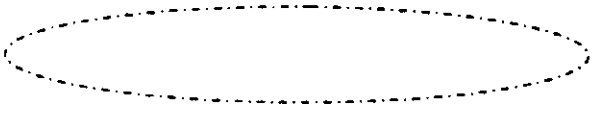


Figure 4 : Cas d'utilisation

-Classe active : c'est une classe dont les objets possèdent un ou plusieurs processus ou threads et qui peut lancer une activité de commande. Elle est différente d'une classe dans le sens où ses objets représentent des éléments dont le comportement est différent de celui des autres éléments. Elle à la même forme qu'une classe mais avec un trait épais.

-Composant : c'est une partie physique remplaçable d'un système, qui se conforme à un ensemble d'interfaces et permet la réalisation. Dans un système, il existe différents types de composants de déploiement, tel que le composant COM+ ou java Beans, ainsi que des composants qui constituent les interfacts du processus de développement, tel les fichiers du code source. Généralement, un composant représente l'enveloppe physique d'éléments de nature logique, comme des classes, des interfaces ou des collaborations.

-Un nœud : c'est un élément physique qui intervient lors de la phase d'exécution, il représente une ressource de calcul et dispose généralement au moins d'un peu de mémoire et souvent d'une capacité de traitement.

Ces sept éléments constituent les éléments structurels de bases qui peuvent êtres inclus dans un modèle UML.

Il existe également des variation de ces sept éléments, comme les acteurs, les signaux, les utilitaires (sortes de classes), les processus et les threads (sorte de classe actives), les applications, les documents, les fichiers, les bibeloteurs, les pages et les tables (sortes de composants).

-Eléments comportementaux :

Les éléments comportementaux représentent les parties dynamiques des modèles UML. Ce sont les verbes du modèle et ils représentent son comportement dans le temps et dans l'espace.

Il existe deux types d'éléments comportementaux :

-Une interaction : c'est un comportement qui comprend un ensemble de message échangés au sein d'un groupe d'élément, dans un contexte particulier pour atteindre un but bien défini.

-Un automate à états finis : c'est un comportement qui précise les séquences d'états d'un élément ou d'une interaction au cours de leurs durées de vie, en réponse à des événements, ainsi qu'à leurs réactions à ces événements.

Ces deux éléments constituent les éléments comportementaux de base qui peuvent être inclus dans un modèle UML. Sur le plan sémantique, ces éléments sont lie à divers éléments structurels, essentiellement des classes, des collaborations et des objets.

-Eléments de regroupement

Il représente les parties organisationnelles des modèles UML. Ce sont des boites dans lesquelles un modèle peut être décomposé. Il existe un seul type fondamental d'élément de regroupement : le paquetages.

Un paquetage est un mécanisme général qui permet de regrouper des éléments, des élément structurels, des élément comportementaux, et même d'autres éléments de regroupement peuvent être range dans un paquetage. A l'inverse des composants (il existe lors de la phase d'exécution), c'est un élément purement conceptuel (il existe seulement lors de la phase de développement)

Ajouter aux paquetages qui sont des éléments fondamentaux, il existe des variations, tels que les frameworks, les modèles et les sous-systèmes (sortes de paquetages).

-Eléments d'annotation :

Les éléments d'annotation représentent les partis explicatifs des modèles UML. Ce sont des commentaires qui peuvent accompagner tout élément dans un modèle, à des fins d'explication, de description, et de remarque. Il existe un seul type, appelé "note". Une "note" est simplement un symbole utilisé pour représenter les contraintes et les commentaires rattachés à un élément où a un ensemble d'éléments.

-Relation d'UML

Il existe quatre types de relation dans UML :

-Dépendance : c'est une relation sémantique entre deux éléments selon laquelle un changement apporté à l'un (élément indépendant) peut affecter la sémantique de l'autre (élément dépendant). La dépendance est représenté par une ligne en pointilles qui peut être flèche, elle comprend parfois une étiquette.

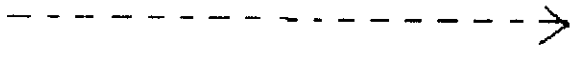


Figure 5 : Relation de Dépendance

-Association : c'est une relation structurelle qui décrit un ensemble de lien, un lien constituant une relation entre différent objets. L'agrégation est un type particulier d'association, qui représente une relation structurelle entre un tout et ses parties.

Elle est représente par une ligne qui peu être fléchée, elle comprend parfois une étiquette et souvent d'autre décoration, comme la multiplicité et les noms de rôles.

Figure 6 : Relation d'association

-Généralisation : c'est une relation de spécialisation/généralisation selon laquelle les attributs de l'élément spécialisé (l'enfant) peuvent se substituer aux attributs de l'élément parent. De cette manière l'enfant partage la structure et le comportement du parent.

Une relation de généralisation est représenté par une flèche dont le trait est plein et dont la pointe creuse est dirigés vers le parent.

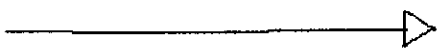


Figure 7 : Relation de Généralisation

-Une réalisation : c'est une relation sémantique entre classificateurs, selon laquelle un classificateurs spécifie un contrat dont l'exécution est garantie par un autre classificateur. Les relations de réalisation apparaissent à deux occasions : entre les interfaces et les classes ou les composants qui les réalisent, et entre les cas d'utilisation et les collaborations qui les réalisent.

Elle est représentée par un mélange d'une agrégation et d'une relation de dépendance.

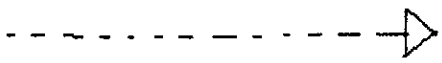


Figure 8 : Relation de Dépendance

En plus de ces quatre éléments, il existe des variations, comme le raffinement, la trace, l'inclusion et l'extension (pour les dépendances).

-Distinctions communes : en UML il existe deux distinctions fondamentales :

-Entre une classe et un objet : Une classe est une abstraction alors que l'objet est la manifestation concrète de cette abstraction.

-Entre interface et implémentation : Une interface définit un contrat et l'implémentation représente la réalisation concrète de ce contrat respectant la sémantique complète de l'interface.

-Mécanisme d'extensibilité :
UML étant un langage ouvert qui peut être élargi pour suffire à exprimer toutes les nuances possibles dans le model, adopte les mécanismes d'extensibilité suivants :

-Stéréotype : il permet de créer de nouvelles briques de base qui dérivent de celle un problème qui existe mais qui sont adapte un problème donne. Par exemple en C++, on travaille avec des exceptions qui ne sont que des classes traites de manière spéciale.

-Etiquette : elle permet la création de nouvelles informations spécifiant un élément en élargissant ces propriétés.

-Contrainte : élargit la sémantique d'une brique de base, permettant l'introduction de nouvelles règles ou en modifiant celle existante.

d-L'architecture :

L'architecture est la clé de voûte du succès d'un développement car elle décrit des choix stratégiques qui déterminent les qualités du logiciel.
La figure est une vue qui a beaucoup inspire UML dans son modèle d'architecture.

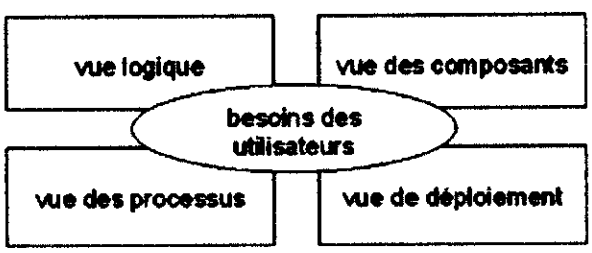


Figure 9 : Modèle d'architecture

- La vue logique : c'est une vue de haut niveau qui modélise les éléments et mécanismes principaux du système, elle se base sur l'abstraction et l'encapsulation. Elle identifie les éléments du domaine, ainsi que les relations et interactions entre ces éléments, et organisent les éléments en catégories.

b-Les règles d'UML

Comme tout langage, UML possède un certain nombre de règles qui permettent de produire des models correctement mis en formes.

Les règles sémantiques d'UML sont :

-Les noms : la manière de designer les éléments, les relations et les diagrammes;

-Le contexte : l'environnement qui donne une signification bien précise à un nom

-La visibilité : la manière dont ces noms peuvent être vus et utilisés par d'autres.

-L'intégrité : la manière dont les objets établissent des relations correctes et cohérentes entre eux.

-L'exécution : les conséquences de l'exécution ou de la simulation d'un modèle dynamique.

On peut aussi trouver des modèles pas correctement mis en forme:

-Partiel : afin de simplifier la représentation graphique, certains éléments peuvent être cachés.

-Incomplets : il manque certains éléments.

-Incohérents : l'intégrité du modèle n'est pas garantie.

Ces modèles incomplets sont inévitables car les détails d'un système se précisent et se combinent au long du cycle de développement du logiciel.

c-Mécanismes généraux d'UML

Quatre mécanismes permettent rendent UML plus simple à utiliser :

-Spécification : dans UML, derrière chaque notation graphique d'un élément, il existe une spécification en fournissant un énoncé textuel de la syntaxe et de la sémantique pour cet élément. Par exemple pour une icône d'une classe on trouve une spécification qui petite les attributs, les opération....

-Décoration : d'autre informations (visibilité de ces attributs et de ses opération, ou encore si elle est abstraite ou non) peuvent être dans la notation d'une classe,
Ceci en les représentant par des décoration graphique ou un textuelle ajoutée.

- La vue des composants : c'est une vue de bas niveau qui montre l'allocation des éléments de modélisation dans des modules (fichiers sources, bibliothèques dynamiques, bases de données, exécutables, etc...), en identifiant les modules qui réalisent (physiquement) les classes de la vue logique, elle montre aussi les contraintes de développement (bibliothèques externes...) et l'organisation des composants, elle montre aussi l'organisation des modules en sous-systèmes.
- La vue des processus : elle est très importante dans les environnement multitâches, elles montre la décomposition du système en terme de processus (tâches), la communication entre ces processus, la synchronisation et la communication des activités parallèles (threads).

- La vue de déploiement : elle est très importante dans les environnements distribués, décrit les ressources matérielles et la répartition du logiciel dans ces ressources.
- La vue des besoins des utilisateurs (vue des cas d'utilisation) : elle guide et unifie toutes les autres, elle justifie l'architecture d'un système informatique, et définit les besoins du client et centre l'architecture du système sur la satisfaction de ces besoins, et conduit à la définition d'un modèle d'architecture pertinent et cohérent.

II. Organisation de l'application [11,12]

II.1. Diagramme de classes

Les diagrammes de classes représentent un ensemble de classes, d'interfaces et de collaborations, ainsi que des relations. Ce sont es diagrammes les plus fréquents dans la modélisation des systèmes oriente objet. Ils représentent la vue de conception statique d'un système.

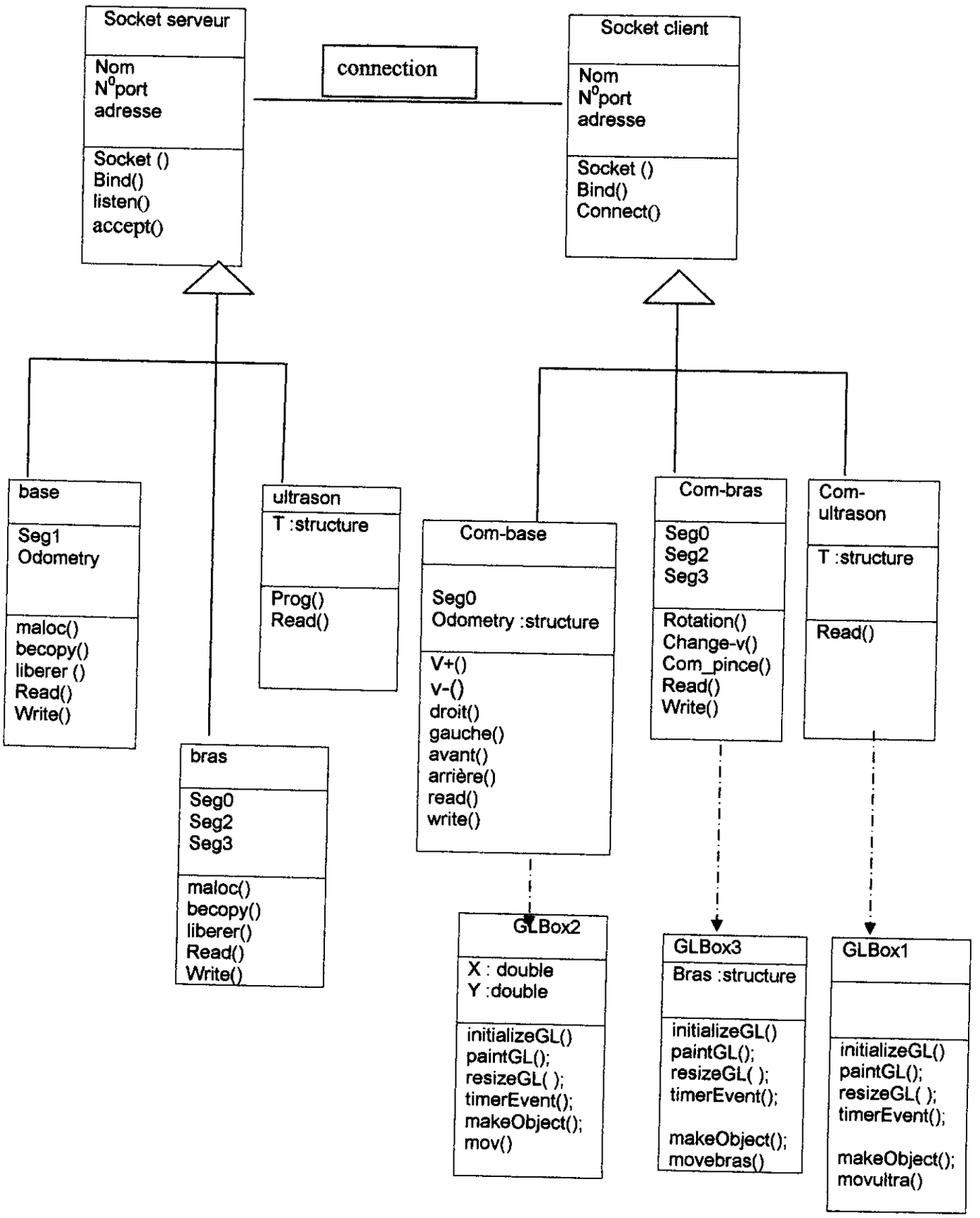


Figure 10 : Diagramme de classes

Seg0, Seg1, Seg2, Seg3 sont des segments mémoire

- Seg0 contient les positions et les vitesses de 6 axes du bras manipulateur
- Seg1 contient:
    1. Les données pour le joystick.
    2. Les deux données pour la vitesse de la roue gauche et droite.
    3. Les deux données pour l'encodeur de la roue gauche et droite.
    4. Un sémaphore/Mutex pour la synchronisation.
- Seg2 contient les données concernant le capteur d'effort et l'état de la pince du bras.
- Seg3 contient les positions courantes des axes.

Nom : c'est le nom de la socket.

N port : c'est le numéro de port.

Adresse : adresse IP.

T : structure contient les valeurs des capteurs ultrason.

Odometry : structure contient les positions du robot(X et Y et θ).

II.2. Diagramme d'objet

Les diagrammes d'objets représentent un ensemble d'objets et leurs relations. Ce sont des vues statiques des instances qui apparaissent dans le diagramme des classes.

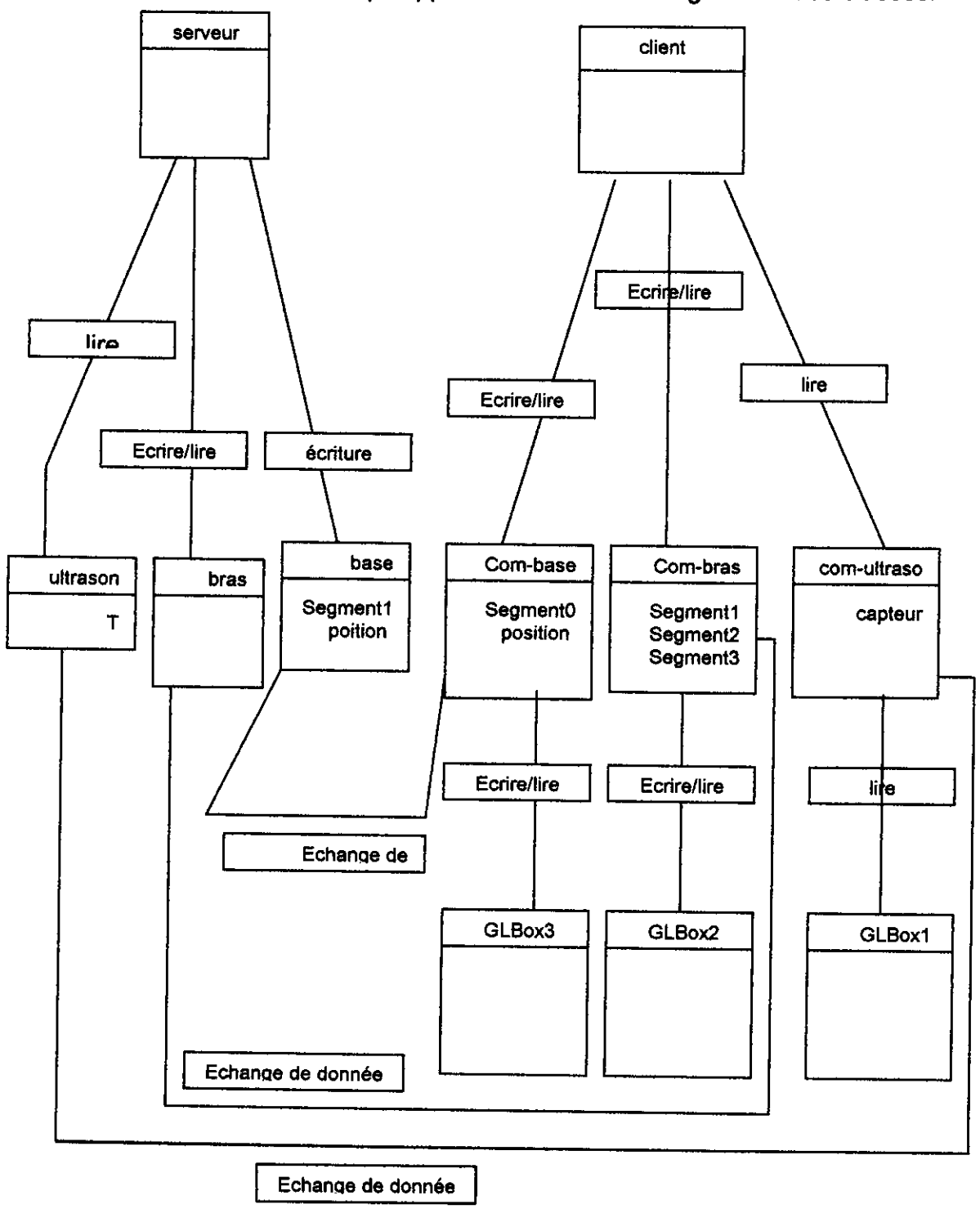


Figure 11 : Diagramme d'objet

Au niveau du serveur, on a créé une instance pour chaque classe (base, bras et ultrason) avec les paramètres suivants :

- Adresse IP : 10.1.4.157
- Port : 9734, 9735 et 9736 pour les trois classes.

Concernant le client, en plus des trois instances des classes (com-base, com-bras et com-ultrason) qui ont les mêmes paramètres que ceux des classes du serveur, on crée les instances des trois classes d'affichages des données (GLBOX1, GLBOX2 et GLBOX3).

II.3. Diagramme de cas d'utilisation

Les diagrammes de cas d'utilisation représentent un ensemble de cas d'utilisation et d'acteur (sorte de classe particulière) et leurs relations. Ils sont particulièrement importants dans l'organisation et la modélisation des comportements d'un système.

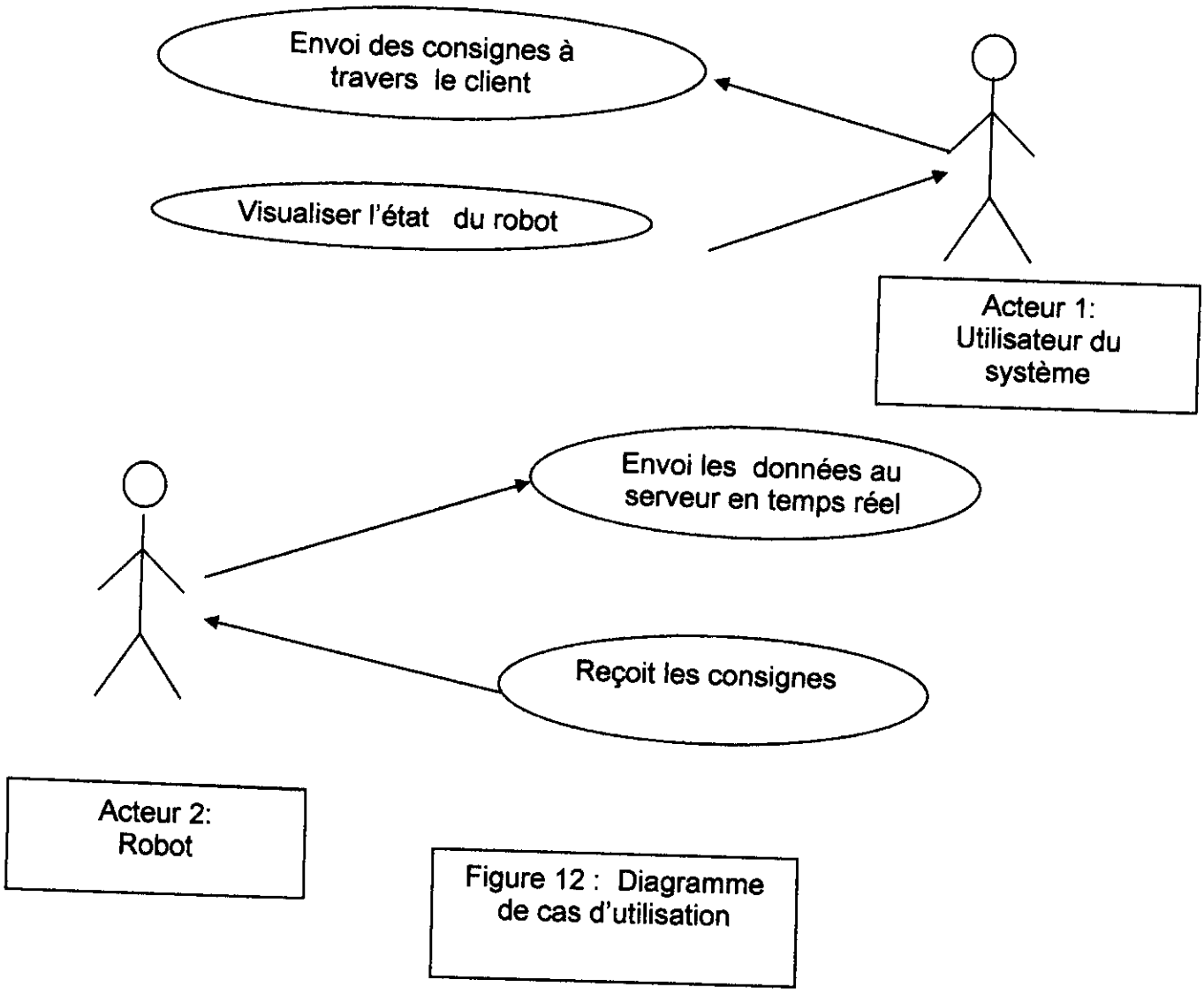


Figure 12 : Diagramme de cas d'utilisation

On peut définir deux acteurs (utilisateurs externes) :

- L'opérateur (humain) : il a la possibilité de visualiser l'état du robot en temps réel et d'envoyer des consignes au robot à travers le client de notre système.
- Le Robot : le robot interagit avec le serveur de notre système à travers la mémoire partagée (Linux RTAI). Il peut envoyer et recevoir des données en temps réel.

II.4. Diagramme d'interaction :

Les diagrammes d'interaction représentent une interaction, c'est-à-dire un ensemble d'objets et leurs relations, ainsi que les messages qu'ils peuvent s'échanger. Ils représentent la vue dynamique du système.
Il y a deux types de diagrammes d'interaction :

II.4.1. Diagramme de séquence

Les diagrammes de séquences sont des diagrammes d'interactions qui mettent l'accent sur le classement chronologique des messages.

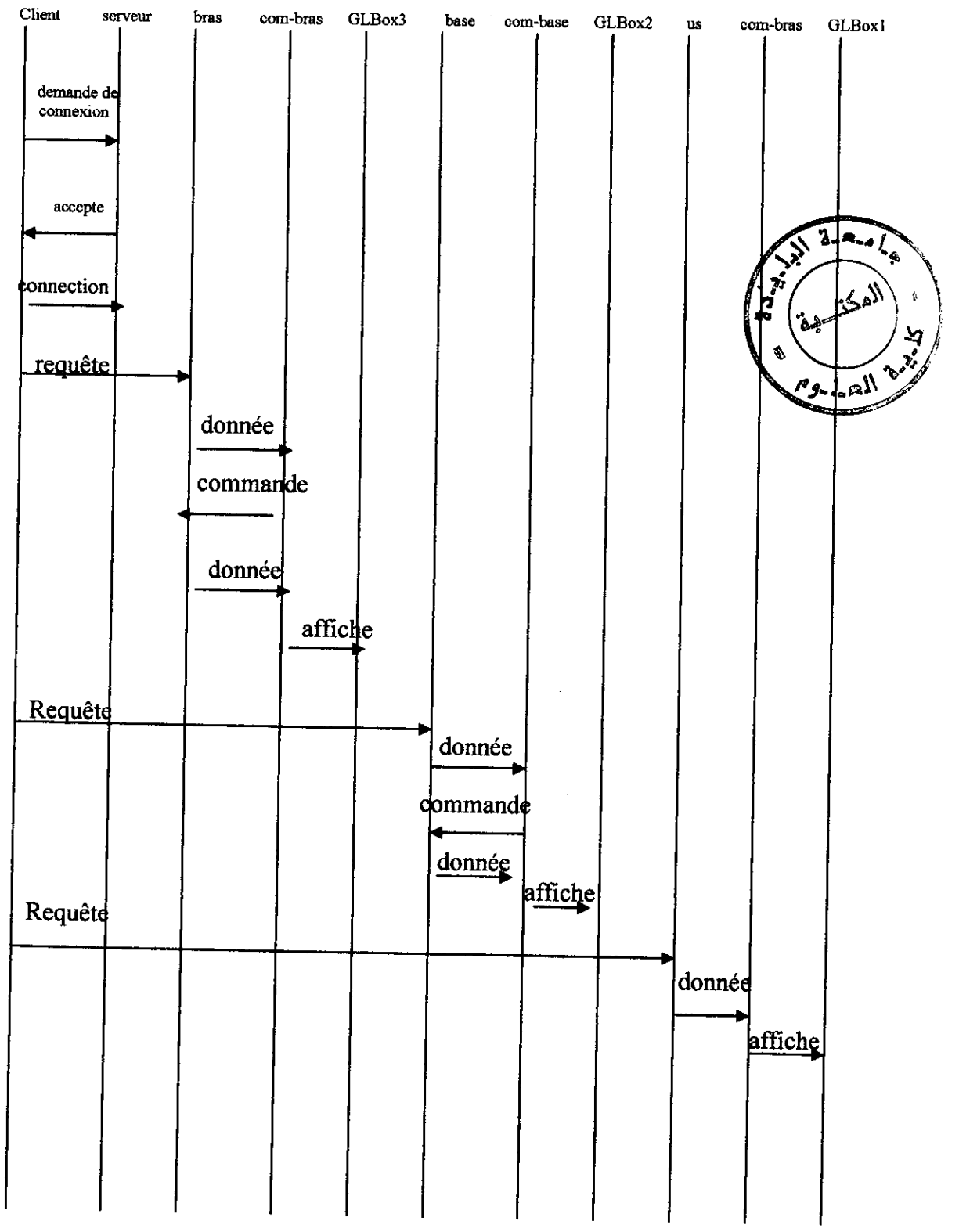


Figure 13 : Diagramme de séquences

Au lancement du système, le client établie une connexion avec le serveur, sans laquelle aucune autre séquence d'utilisation n'est possible.
Après le lancement du système (client et serveur), il existe trois séquences possibles d'utilisation de notre système :

- Envoi de requête à la base mobile.
- Envoi de requête au bras manipulateur.
- Réception des données des capteurs ultra son.

II.4.2. Diagramme de collaboration

Les diagrammes de collaborations sont des diagrammes qui mettent l'accent sur l'organisation structurelle des objets qui envoient et renvoient des messages.

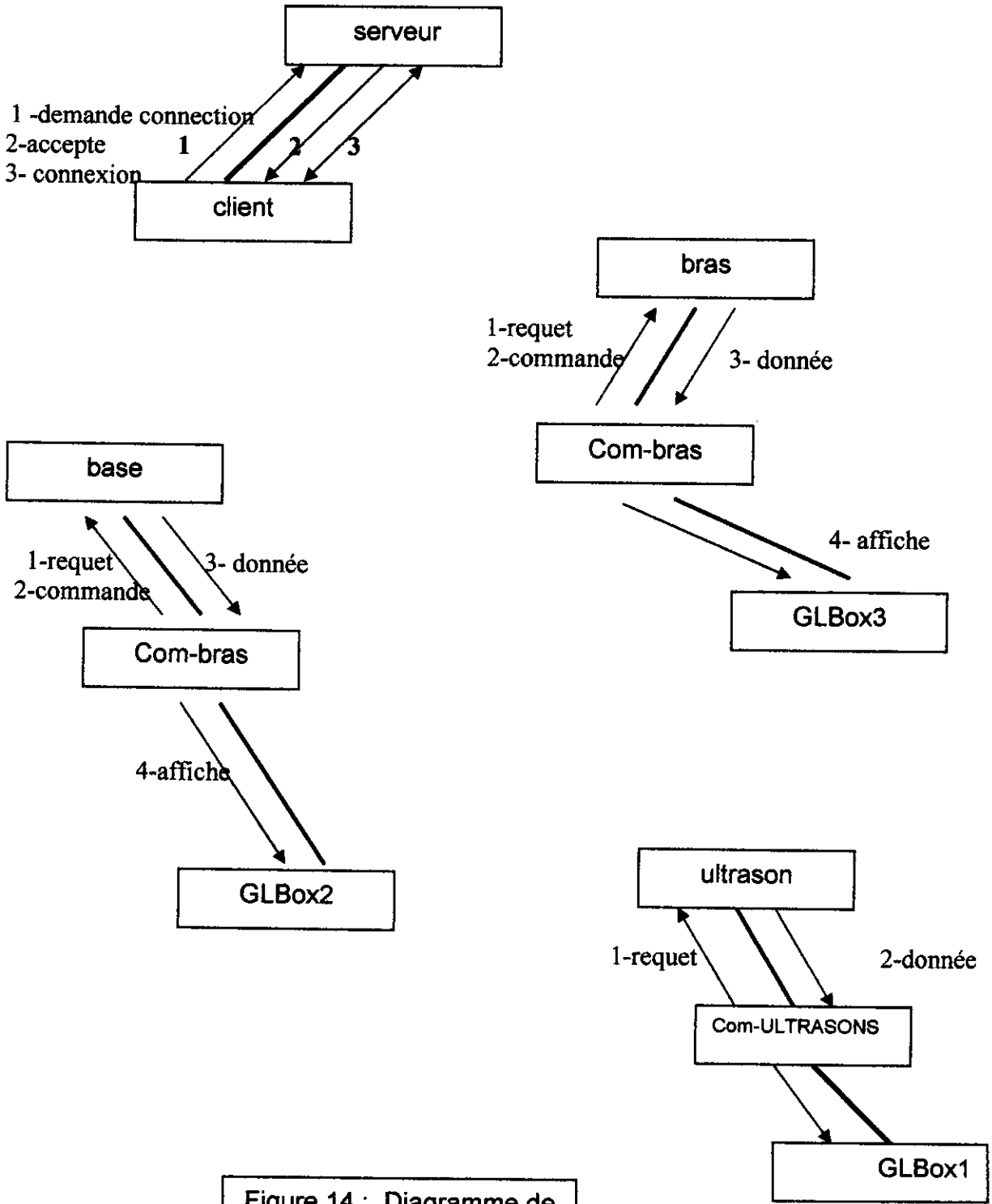


Figure 14 : Diagramme de collaboration

II.5.Diagramme d'activité

Les diagrammes d'activité sont un type particulier du diagramme d'états-transitions qui décrit la succession d'activités aux seins d'un système. Ils représentent une vue dynamique du système.

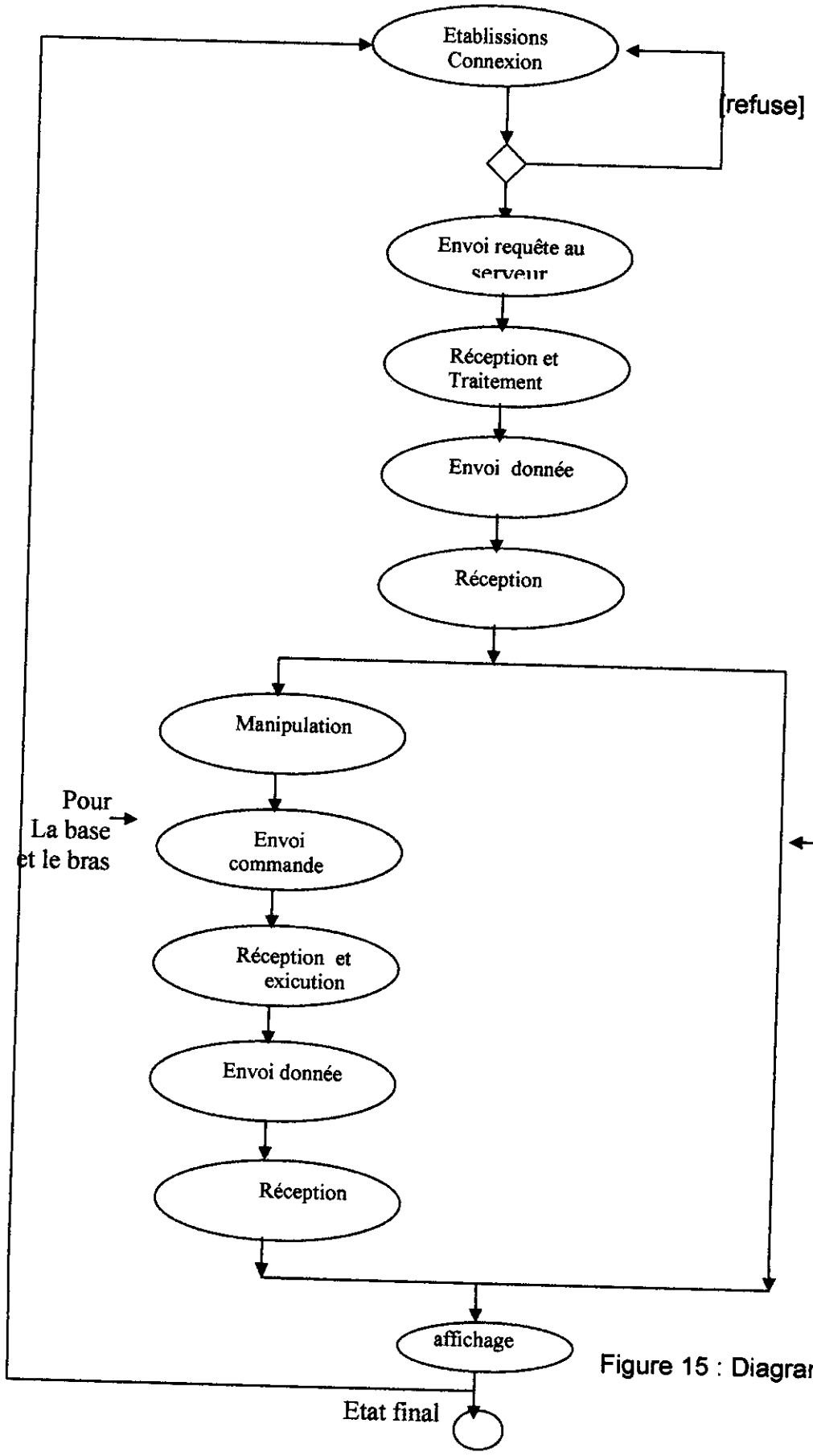


Figure 15 : Diagramme d'activités

CHAPITRE V

IMPLEMENTATION ET EXPERIMENTATION

I. Introduction

Ce chapitre est consacré à la présentation de notre application qui consiste à commande le robot manipulateur mobile (Robuter-ULM du CDTA) à distance. En premier nous survolerons les différents outils informatiques utilisés dans l'implémentation de notre application pour le contrôle à distance du robot. Pour cela, nous présenterons un aperçu du système d'exploitation linux et de la bibliothèque graphique Open GL, du langage de programmation utilisé qui est le c++ et de Syndex qui est l'environnement de programmation du bas niveau permettant la commande du robot.

II. Environnement de développement [4,13]

II.1. système d'exploitation

Nous avons travaillé pour notre application sous système d'exploitation linux (Red Hat 9.0) vu que le Pc Embarque du robot tourne sous linux (RedHat 6.2). Linux est un système fiable, stable et pas cher.

La contrainte de la gestion du temps réel dans la manipulation du robot manipulateur mobile, qui demande un bon asservissement qui garantie à chaque instant la stabilité de la position, ainsi que la mise a jour de certaines variables d'états, nous ramène à l'utilisation de la couche RTAI de linux (RTAI 1.3). RTAI est une couche intermédiaire entre la couche bas niveau et la couche haut niveau. Le noyau temps réel tourne au niveau le plus proche du hardware. Un programmeur gère une série de taches en temps réel avec un accès total au hardware, linux, lui-même est vu par le programmeur comme une tache temps réel.

Le système RTAI de linux est très utilise par les professionnels de l'automatisme du fait que c'est un système qui utilise les ressources hardware correctement, très stable, son coût est faible et parfois nul, le code source est facile a modifier et a enrichir du fait qu'il est public, et il présente un grand nombre de documentation.

II.2. Méthodologie AAA

La méthodologie AAA (Aadequation Algorithme-Architecture) fondée sur des modèles de graphes, pour spécifier les algorithmes applicatifs et les architectures distribuées, et pour déduire les implantations possibles en termes de graphes.

II.3. Syndex

Syndex est u logiciel de CAO (Conception Assiste par Ordinateur) niveau système, fondé sur la méthodologie AAA pour le prototypage rapide et l'optimisation d'applications distribuées temps réel embarquées complexes. Il permet de spécifier à l'aide de graphes les algorithmes applicatifs et les architectures distribues, de lancer des heuristiques d'optimisations de l'implantation des algorithmes sur les architectures conduisant à une simulation temporelle de l'exécution temps réel, et

enfin de générer automatiquement des exécutifs distribues temps réel sans interblocage et à faible surcoût.

II.4. Le langage C++ :

Le langage choisi pour notre application est c++ du fait de sa compatibilité avec le c qui est le langage qui fait marcher le robot, sauf que pour le cote serveur de notre application nous avons gardé le c pour cause de compatibilité avec le bas niveau. C++ un langage très puissant (c'est un langage orienté objet) et rapide d'exécution, et qui a nombres d'autre atout : grand nombres de fonctionnalités, portabilité des fichiers sources...etc.

II.5. OPEN GL

Open GL (Open GRAPHIQUE Library ou bibliothèque graphique ouverte) est une bibliothèque de programmation graphique développée depuis 1992 par la société Silicom Graphics pour ses stations graphiques hautes performances.
Aujourd'hui elle est devenue un standard industriel, elle a été implantes sur la plupart des plates-formes informatiques actuelles (linux, Windows, MacOS...), elle est disponible pour de nombreux langages de programmation (C, C++, JAVA, DELPHI..).

Open GL est une interface programme pour le hardware graphique. Elle permet de créer un programme interactif qui produit des images couleurs objet en mouvement. Ainsi il permet de contrôler la technologie des cartes graphiques afin de produire des images réaliste en trois dimensions, le langage étant constitue d'un ensemble de fonction permettant de créer un environnement 3D.
Open GL compte 120 commandes et couvre la création et l'animation d'objet 2 et 3D.

Voici en résume les principales possibilités d'Open GL:

- Primitives géométriques : elles permettent de construire des descriptions mathématiques d'objets a partir de points, lignes, polygones, images.
- Codage de couleur : en mode RGBA (Rouge, Vert, Bleu, Alpha) ou en mode index de couleurs.
- Visualisation et modelage : permettent d'arranger l'objet dans une scène tridimensionnel, de bouger les cameras dans un espace et de choisir le point de vue d'où sera visualisée la scène.
- Projection de texture : apporte du réalisme au modèle en améliorants l'aspect des surfaces des polygones.
- Eclairage matériaux : c'est une partie indispensable de tout infographe 3D, Open GL fournit la commande pour calculer la couleur de n'importe quel point à partir des propriétés des matériaux et des sources lumières dans la pièce.
- Transformation : rotation, translation, perspectives en 3D.

Open GL comprend aussi quelques difficultés et insuffisances, ainsi c'est un langage assez complexe, mais qui ne nécessite pas de connaissances en programmation. Aussi il a un problème avec les fonctions qui ne sont pas gérées par l'OpenGL comme la création de la fenêtre de l'application par exemple. Ce dernier problème est résolu par l'intervention de la GLUT (Graphics Library Utility ToolKit), qui est en fait une "boite à outils" compatible avec tous les OS sous lesquels OpenGL fonctionne. Cette librairie simplifie grandement la tâche du programmeur, puisqu'elle est dédiée aux entrées-sorties, la création de la fenêtre,...etc.

III. Présentation de l'application

Construire un système de commande à distance d'un robot manipulateur mobile (figure 1) mène toujours à une interaction des capteurs et des actionneurs avec le logiciel responsable du contrôle. Les capteurs scannent l'environnement du système et offrent les caractéristiques requises au logiciel. En utilisant ces données, l'opérateur est en mesure de percevoir une image de l'environnement réel et peut réagir en conséquence. La figure 1 montre le but à atteindre.

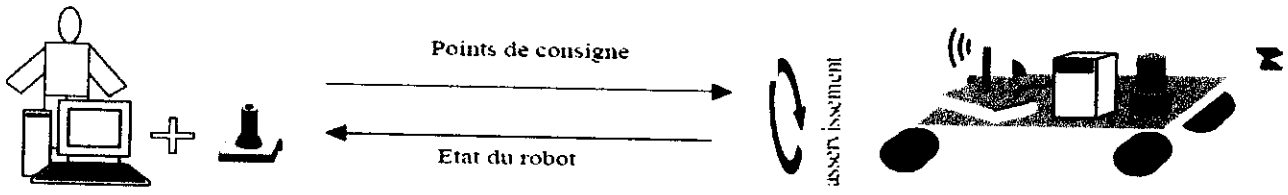


Figure 1 : Commande à distance du Robuter-ULM

La commande à distance (figure 2) réalisée repose sur une architecture client/serveur :

- Le serveur tourne sur le PC embarqué du robot manipulateur mobile et traite les requêtes qui lui parviennent du client. De ce fait, le serveur permet à l'opérateur d'accéder aux ressources matérielles du robot (actionneurs et capteurs),

- Le client est un programme exécuté sur un PC distant. Il permet à travers une interface à l'opérateur de percevoir l'état du robot ainsi que d'envoyer les commandes souhaitées.

Par conséquent l'application doit prendre en charge :

- La gestion des communications entre les structures de commande du robot (PC embarqué du robot) et PC Hôte (client).
- possède une interface utilisateur susceptible de contrôler simultanément les structures du robot (la base mobile, bras manipulateur)
- permette de contrôler cet ensemble via le réseau informatique local (modèle client/serveur).

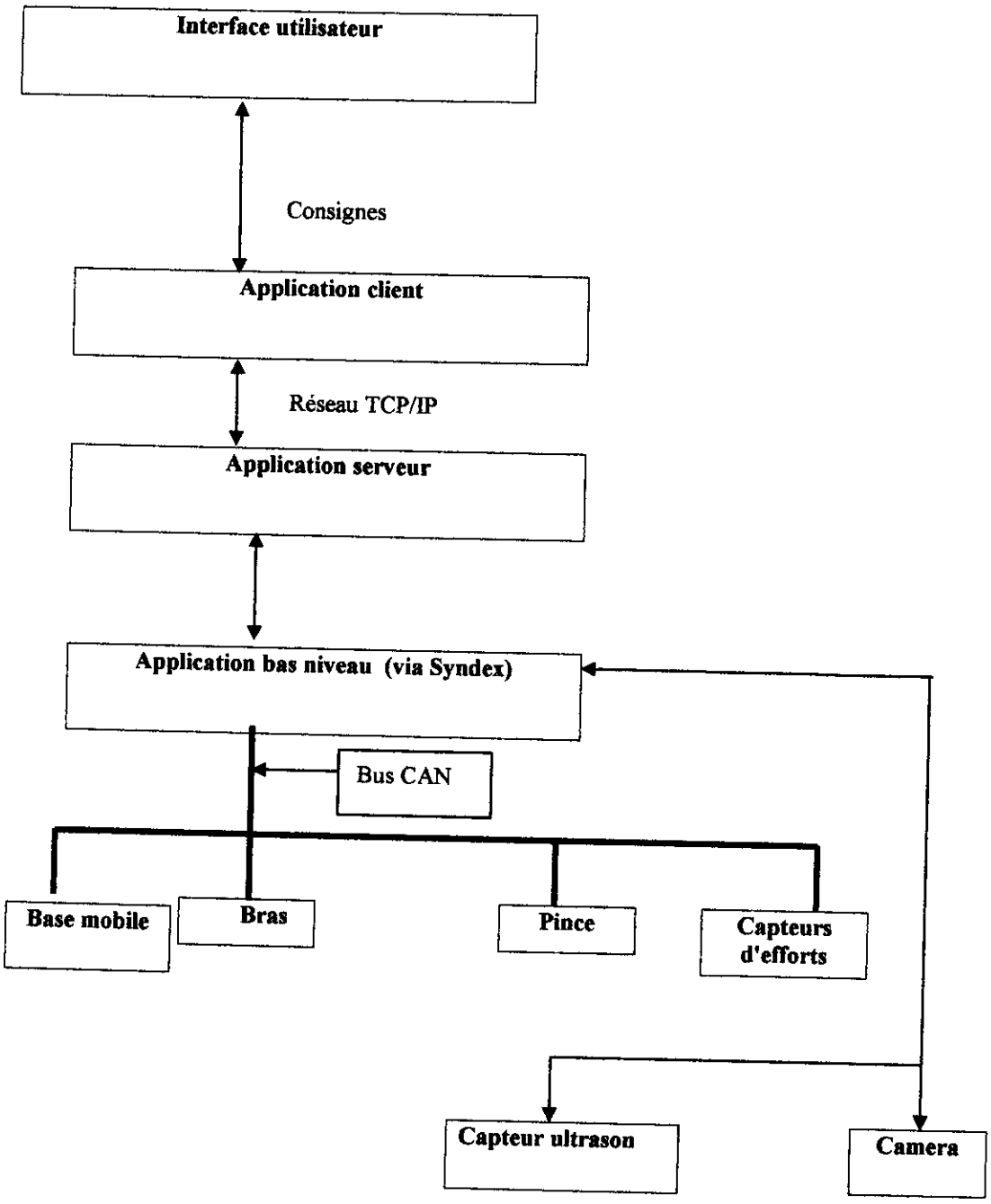


Figure 2 : Présentation de l'application

III.1. Architecture Client/Serveur

La première étape dans le contrôle à distance du robot est l'établissement de la connexion entre le robot (serveur) et le pc hôte (client). Notre application est basée sur une architecture client/serveur qui utilise le mécanisme des sockets. La figure 2 montre le fonctionnement des sockets en mode connecté :

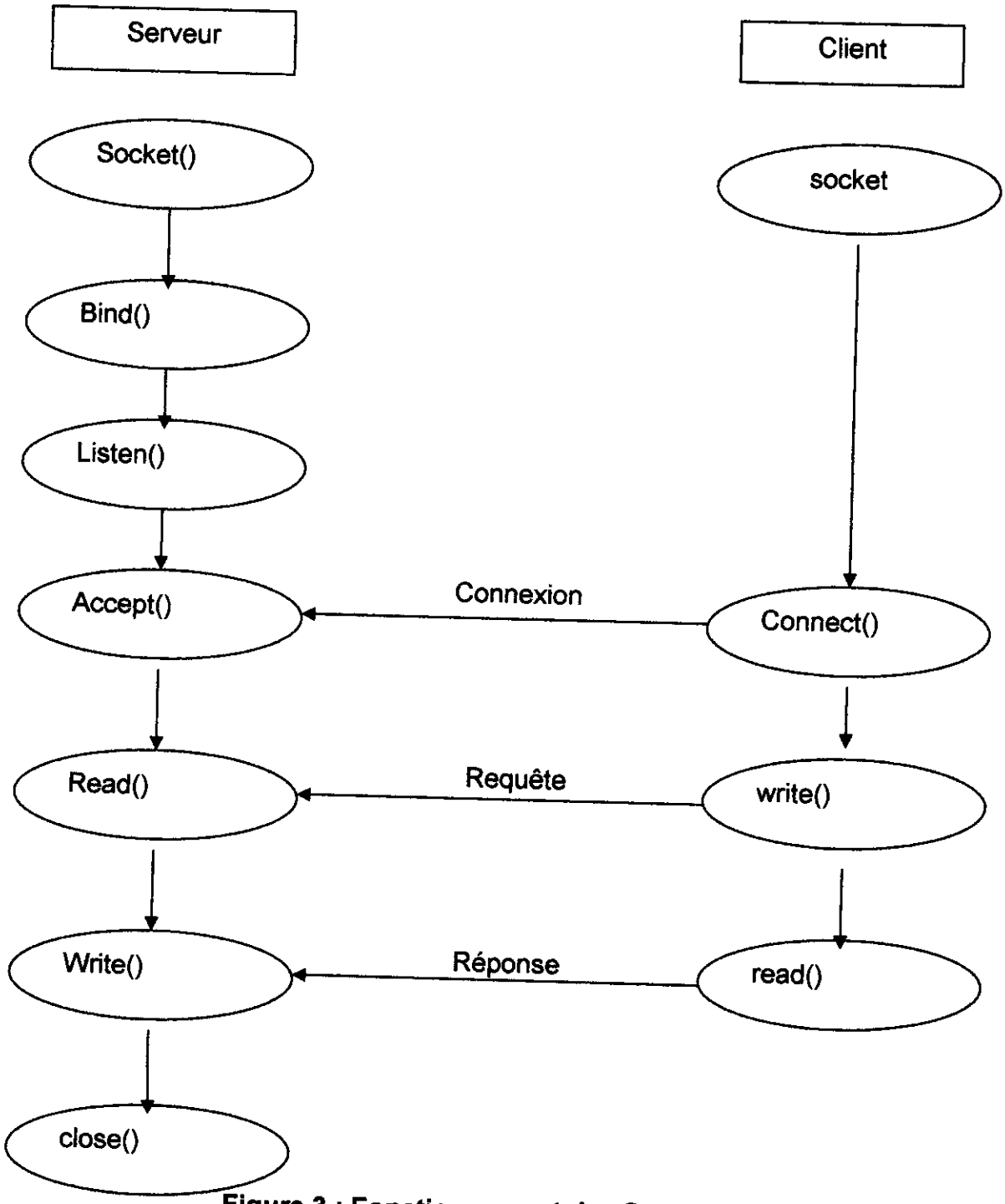


Figure 3 : Fonctionnement des Sockets en Mode connecté

Après l'établissement de la connexion entre le client et le serveur le client pourras avoir accès aux différentes ressources du robot et ainsi de le contrôler.

III.2. La base mobile

Notre application offre à l'opérateur toute les commandes pour piloter la base mobile en mode manuel en lui donnant accès à tous les degrés de liberté de la base mobile. Par conséquent, l'opérateur possède la maîtrise totale de la direction moyenne du mouvement, et d'évitement des obstacles sur le chemin. Pour avoir une image de l'état de la base mobile le serveur transfert une copie de la mémoire partagée au client. Ce segment de mémoire contient les informations concernant la vitesse de chaque roue et les valeurs codeur de chaque roue. De ce fait l'opérateur a la possibilité de changer de direction et/ou de vitesse ce qui a pour effet de modifier les données sur les vitesses de chaque roue du robot. La copie du segment de mémoire ainsi modifié et retransmise au serveur qui permet la mise à jour de la copie du segment de mémoire de son coté puis le transfert au bas niveau via la couche RTAI. Ceci a pour conséquence l'exécution de l'action demandée.

La figure 3 illustre le fonctionnement de la commande de la base mobile où :

(1) : Lecture de la mémoire partagée provenant de bas niveau par le serveur.

(2) : Envoi de la copie de cette mémoire au client à travers la socket.

(3) : Envoi de la copie de la mémoire après manipulation par le client vers le serveur via la même socket

(4) : Envoi de la copie du segment de mémoire du serveur vers le bas niveau (exécution de la commande).

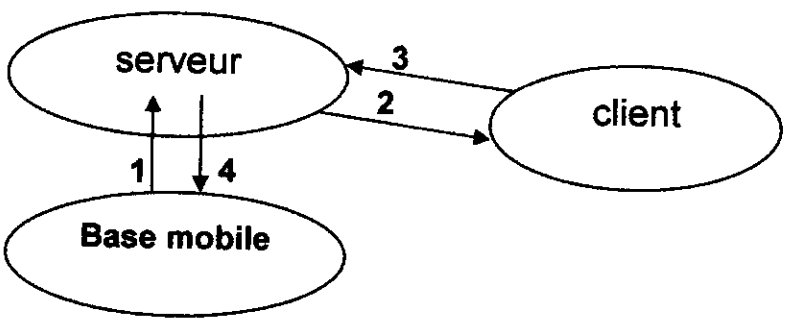


Figure 4 : Commande de la base mobile

L'algorithme du mécanisme d'échange de données entre le bas niveau et le serveur via la mémoire partagée se résume comme suit :

1. Allocation de la mémoire pointée par un segment,

2. Si la mémoire est occupée aller à (7),
3. Lecture des données provenant du bas niveau,

Après Réception des données provenant du client :

4. Si la mémoire est occupée aller à (7),
5. Le serveur Ecrit dans la mémoire partagée,
6. Libérer le segment mémoire,
7. Fin

III.3. Les capteurs ultrasons

La ceinture de 24 capteurs ultrasons nous permet d'avoir une couverture de l'environnement proche de la base mobile et de détecter les éventuels obstacles dans périmètre de 3 mètres. Le serveur récupère les données provenant des différents capteurs via une liaison RS232 puis les interprètent en distances. Ces valeurs sont ensuite regroupés dans une structure qui envoyée au client via une socket. De son coté le client récupère ces valeurs puis effectue une représentation graphique de la base mobile ainsi que des obstacles se trouvant dans l'environnement du robot. La figure 4 montre le mécanisme de transfert des données des capteurs ultrason où :

(1) : Le serveur lit les données des capteurs ultrasons

(2) : Le serveur envoi les données du capteur au client via une socket.

(3) : Le client interprète les données et visualise les obstacles

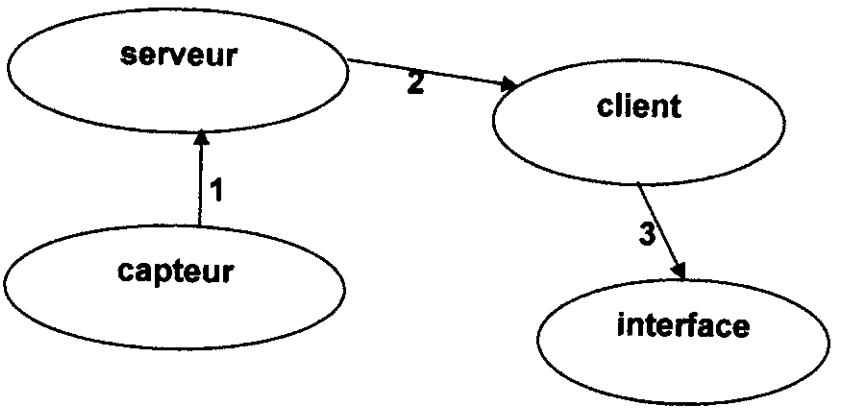


Figure 5 : contrôle des ultrason

III.4. Le bras manipulateur

Le bras manipulateur possède six degrés de liberté ce qui offrent à l'opérateur la possibilité de positionner l'organe terminal du robot comme il le désire (bien sûr dans son volume de travail). A son extrémité le bras possède une pince électrique commandée en tout ou rien qui permet à l'opérateur de manipuler des objets. Le bras et aussi équipé d'un capteur d'effort qui permet à l'opérateur de connaître les efforts exercés par le bras sur son environnement et l'inverse.

L'échange de données s'effectue avec le même principe mentionné dans le cas du contrôle de la base mobile, sauf que pour le bras manipulateur les informations échangées concerneront la position et la vitesse de chaque articulation du bras ainsi que les données provenant du capteur d'effort. Pour cela nous utiliserons trois segments de mémoire : le premier contient les informations sur les positions et les vitesses désirées des 6 axes du bras manipulateur. Le deuxième contient informations sur les positions actuelles des six axes. Le dernier segment contient les données concernant le capteur d'effort et l'état de la pince du bras. La figure 6 montre le mécanisme d'échange de données pour le bras et le capteur d'effort comme suit :

(1) : Le serveur lit les données du bas niveau, ces donnes sont les positions des six axes et les valeurs des forces et couples exercés par le bras.

(2) : Le serveur envoi une copie de cette mémoire au client.

(3) : Le client envoi au serveur les postions et les vitesses désirées ainsi que la commande de la pince.

(4) : Le serveur mis à jour les mémoires partagées puis le bas niveau exécute l'action associé.

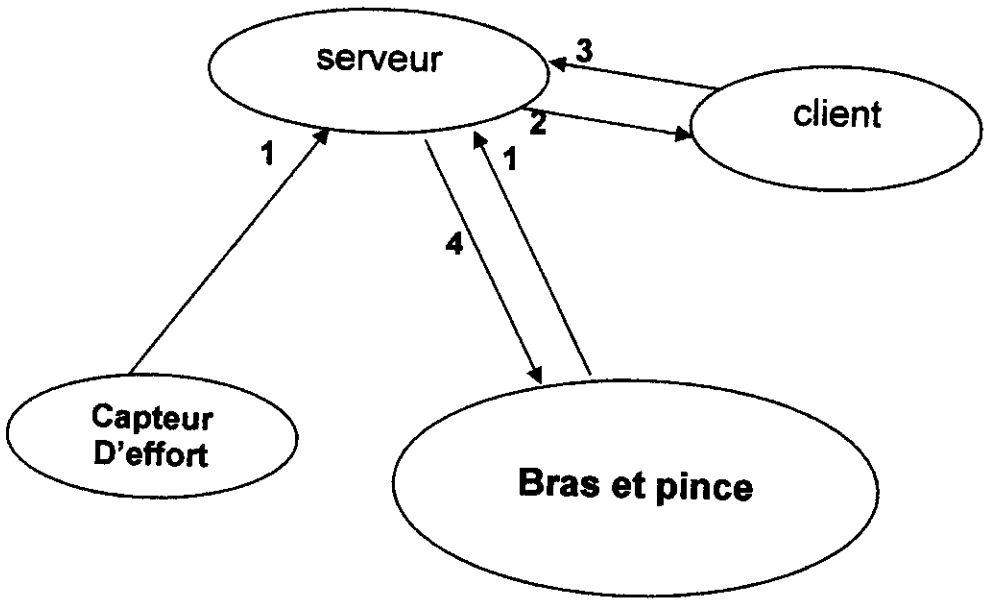


Figure 6 : Commande du bras manipulateur

III.5. Utilisation de L'interface :

Notre application possède une interface ergonomique et facile à utiliser. Cette interface permet, d'un coté, l'affichage des données qui proviennent du serveur et qui offrent à l'opérateur une image de l'état du robot. D'un autre coté, elle donne la possibilité de manipuler le robot à travers des commandes qui se trouvent sur l'interface.

L'interface a été conçue sous QTDesigner qui est un outil de programmation sous linux. En utilisant quelques fonctions de la bibliothèque graphique Open GL, nous avons crée des animations pour montrer l'état du robot.

L'interface comme le montre la (figure7) est subdivisée en 6 sections : Une pour l'interprétation les valeurs des capteurs ultrasons, une autre pour la localisation du robot dans la salle en utilisant l'odométrie, une troisième pour l'affichage du mouvement du bras, Une quatrième pour commander la base mobile, une cinquième section pour commander de bras manipulateur et enfin une dernière qui affiche les valeurs des qui proviennent du capteur d'effort.

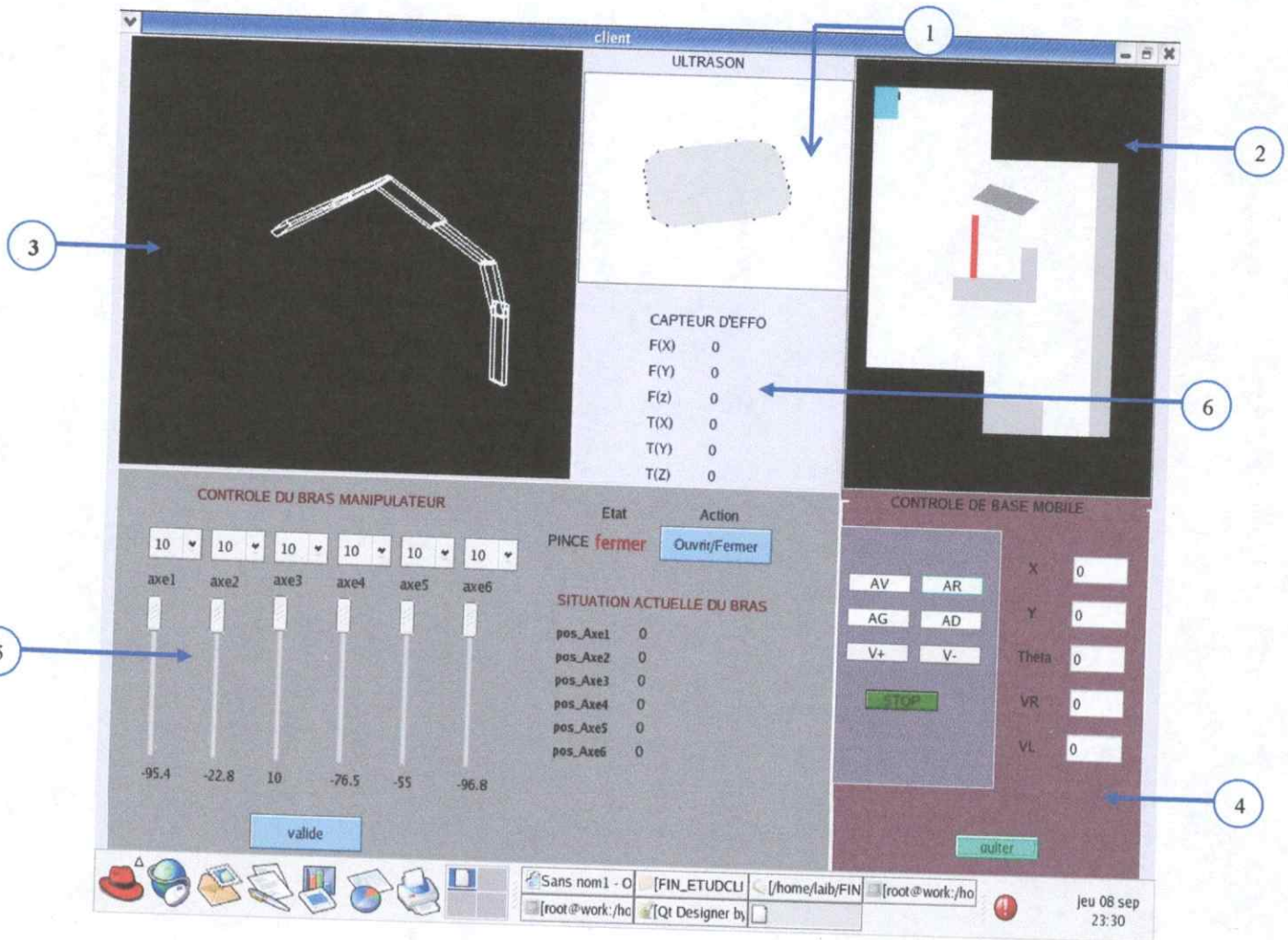


Figure 7 : Interface utilisateur

(1) : le panneau d'affichage interprétation les valeurs des capteurs ultrasons.

(2) : le panneau d'affichage de la salle et le mouvement de robot.

(3) : le panneau d'affichage les mouvements des axes.

(4) : le panneau de contrôle du robot.

(5) : le panneau de contrôle du bras.

(6) : le panneau d'affichage des valeurs du capteur d'effort.

- **Le panneau d'affichage d'interprétation des valeurs des capteurs ultrasons**

Ce panneau (figure 8) permet de modéliser la plate-forme mobile et les obstacles détectés par les capteurs ultrasons du robot. Cet affichage ce fait a l'aide d'une de la bibliothèque Open GL, qui nous permet de dessiner le robot, les capteurs placés sur le robot ainsi que des point rouge représentant les obstacles, et c'est a l'aide de ces point que nous pouvant interpréter la distance qui sépare la base mobile de l'obstacle.

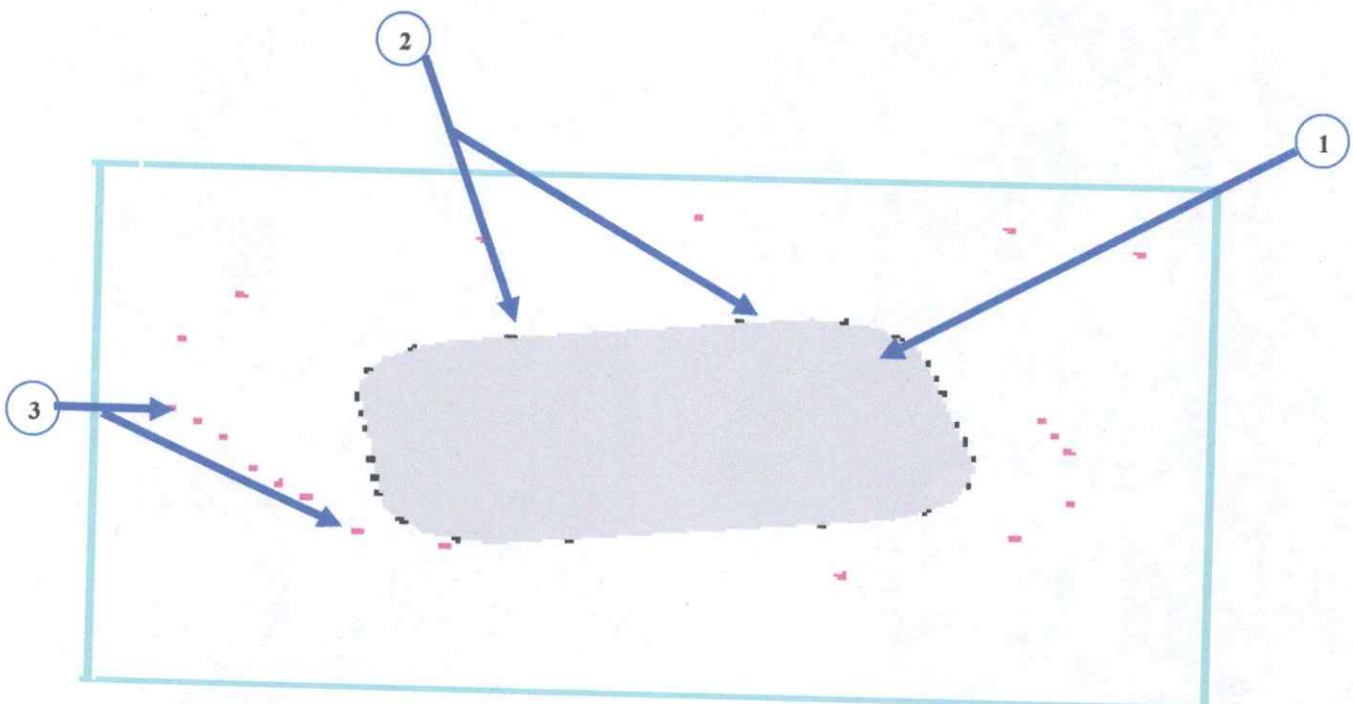


Figure 8 : Panneau d'affichage d'interprétation des valeurs des capteurs ultrason

(1) : représente la base mobile.

(2) : represent l'emplacement des capteurs.

(3) : représente les obstacles.

- **Le panneau d'affichage de la salle et le mouvement de robot (localisation)**

Dans notre application il est nécessaire que l'opérateur puisse avoir un aperçu sur l'emplacement du robot dans la salle où il évolue. Pour cela nous avons conçu une représentation de la salle ainsi que celle du robot. Cette représentation utilise l'odométrie pour localiser le robot dans son environnement d'évolution. L'odométrie est une fonction faisant partie de l'application serveur. Elle fourni la position et l'orientation (X et Y et θ) de la plateforme mobile en se basant sur les informations obtenues à partir des codeurs des deux roues motrices gauche et droite. Il est à noter que l'odométrie nous permet d'obtenir une localisation relative par rapport d'un repère lié à la base mobile. Pour palier à cet inconvénient nous avons eu recours à ce que le point de départ du robot soit toujours le même. Cette solution nous permet de lié repère de la base à un point dans la salle ce qui fourni en définitif une position de la plateforme par rapport au repère fixe de notre salle.

Ce panneau (figure 8) est une représentation du robot en mouvement dans la salle, ainsi que la représentation des différents obstacles fixe qu'il peut rencontrer dans la salle durent son mouvement. Il est à noter que nous ne prenons pas en charge la représentation d'éventuel obstacle mobile. Cette dernière est prise en charge par le panneau des capteurs ultrasons. Cette représentation a été réalisée en faisant recours à la bibliothèque Open GL qui nous a permis de modélisé notre environnement en un modèle réaliste de la salle, des obstacles, et une représentation dynamique du robot en montrant le robot en mouvement réel dans la salle, et c'est cela qui va nous aider pour la localisation du robot.

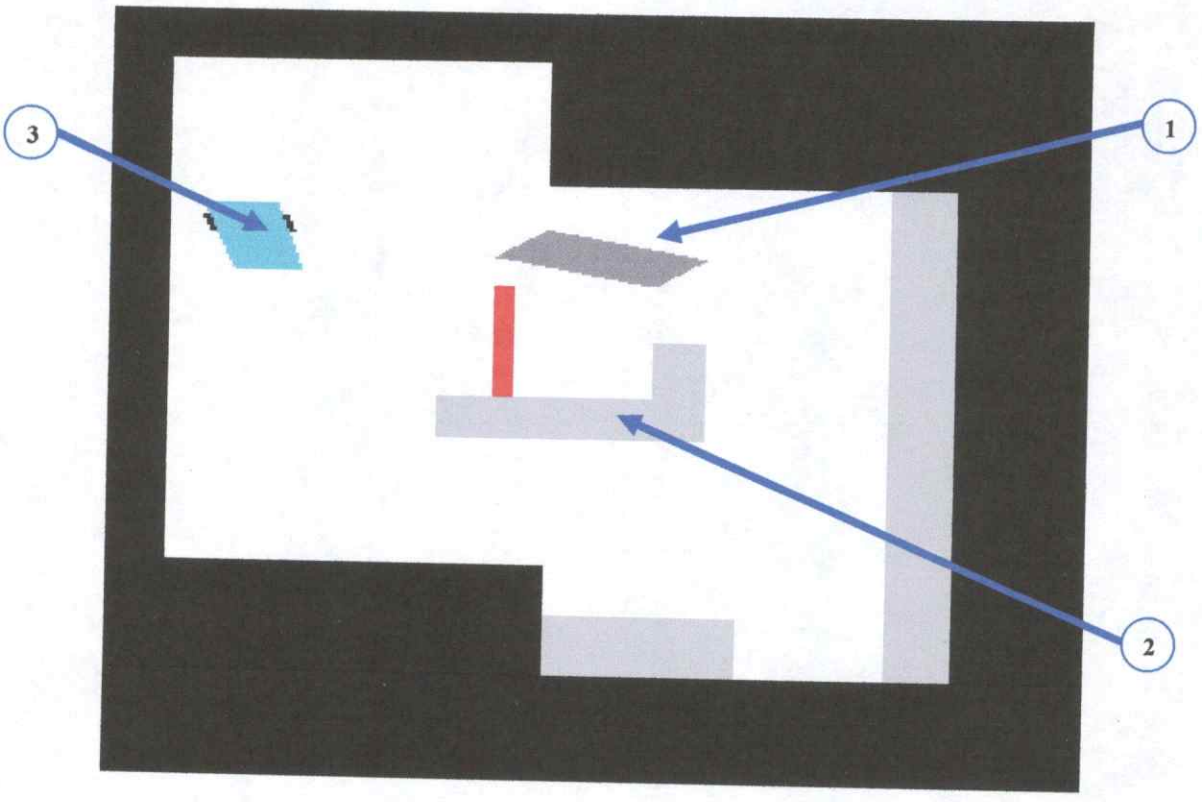


Figure 9 : le panneau d'affichage de la salle et le mouvement de robot

(1) : La salle.

(2) : Les obstacles.

(3) : Le robot en mouvement.

- **Le panneau de contrôle de la base mobile**

Le panneau de contrôle de la base mobile a été réalisé en utilisant l'outil de programmation QTDesigner. Nous avons réalisé une interface avec des boutons pour des commandes qui permettent à l'opérateur de choisir un sens d'orientation de la plateforme ainsi que la vitesse de déplacement. Le panneau contient sept boutons pour le déplacement vers l'avant, le déplacement vers l'arrière, le braquage à gauche, le braquage à droite, l'arrêt du véhicule, l'incrémentation et la décrémentation de la vitesse. Ces commandes sont récapitulées le tableau suivant :

Tableau 1 : Action des boutons

Bouton	Action
AV	Passer en marche avant
AR	Passer en marche arrière

AG	Tourner a gauche
AD	Tourner a doit
V+	Augment la vitesse
V-	Décrémenter la vitesse
STOP	Arrêter progressivement

Afin que l'utilisateur puisse comprendre les données affichées de la base, nous apportons certaines précisions :

- La direction du Robuter est donnée par le différentiel de vitesse des deux roues.
- On marche avant la vitesse de la roue gauche et droit doit être positif.
- On marche arrière la vitesse de la roues gauche et droit doit être négative.
- Pour l'arrêt progressif on est défini une fonction exponentielle E(t) défini :

  $E(t)=v * e^{-t}$

  Où
  V : la vitesse.
  T : durée de freinage.

Le panneau fourni cinq autres champs qui nous permettent de connaitre en temps réel l'état actuel la base :

- le déplacement selon l'axe x.
- le déplacement selon l'axe y.
- l'orientation de la base thêta.
- la vitesse de la roue gauche.
- La vitesse de la roue droite.

- **le panneau de contrôle du bras**

Ce panneau (figure 10) a été aussi réalisé avec l'outil QTDesigner. Il contient différents boutons qui permettent de commander les différentes fonctionnalités du robot (mouvement des axes, choix de la vitesse de déplacement, action de la pince). Ainsi comme le montre la figure 9, nous pouvons changer la position de chaque axe du bras en précisant la vitesse de déplacement, actionner la pince pour l'ouvrir ou la fermer et obtenir l'état du bras et de la pince en temps réel.

Il est à noter que nous avons rajouté un bouton valide pour valider l'envoi de la position désirée du bras (les six axes). Ce bouton procure une sécurité pour robot contre une mauvaise manipulation ainsi le bras ne bougeras pas au toucher des boutons de changement de position des différents axes.

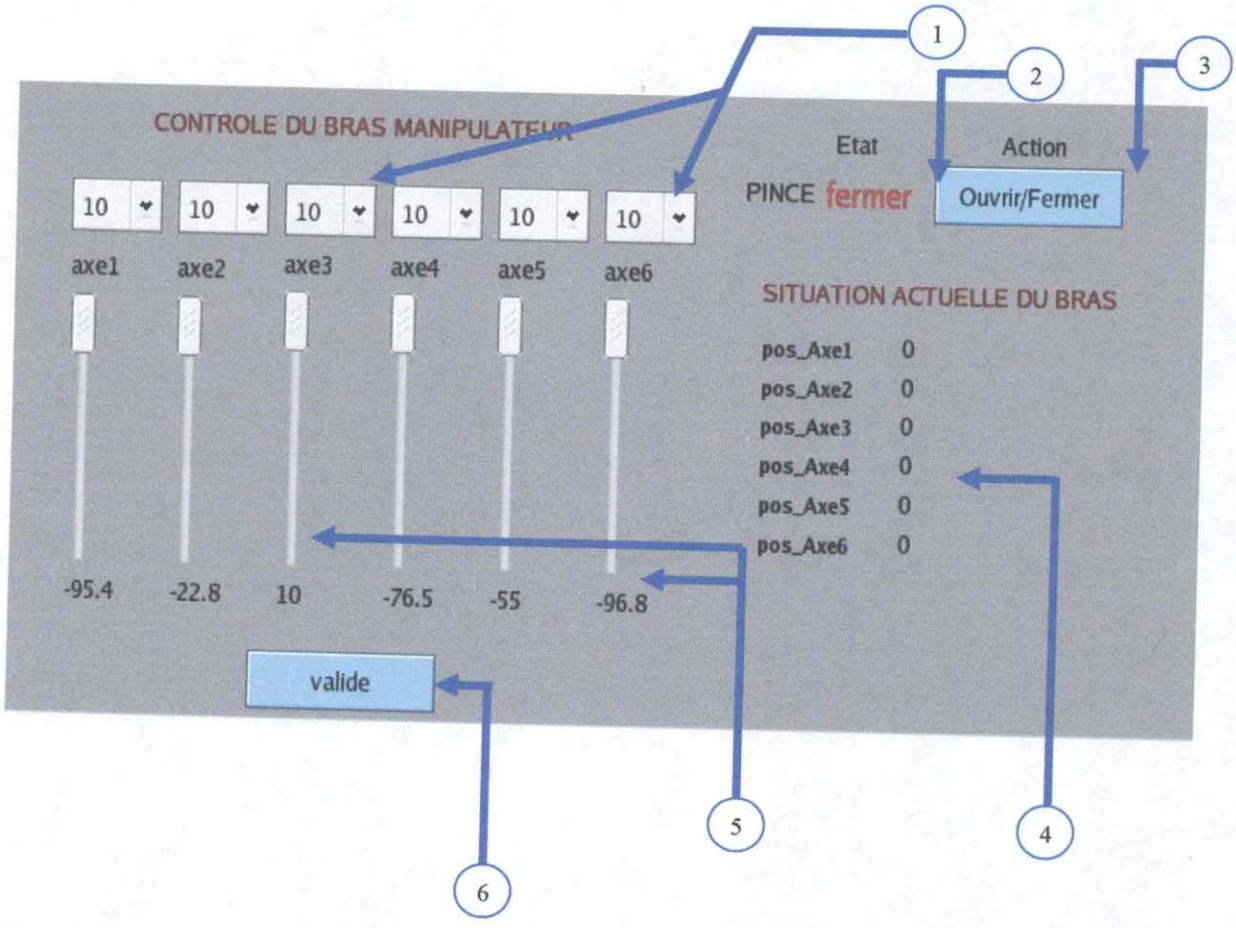


Figure10 : le panneau de contrôle du bras

(1) : pour change la vitesse des axes.

(2) : affichage l'état de la pince (ouvert ou fermer)

(3) : pour commande la pince

(4) : affichage les positions des axes.

(5) : pour la modification les positions des axes.

(6) : pour valide le changement (position et vitesse)

- **le panneau d'affichage les mouvements des axes**

Ce panneau est un complément au panneau précédent et qui fourni à l'opérateur comme le montre la figure 11 une animation de l'état actuel du bras manipulateur. Cette partie a été réalisée en utilisant quelques fonction d'Open GL.

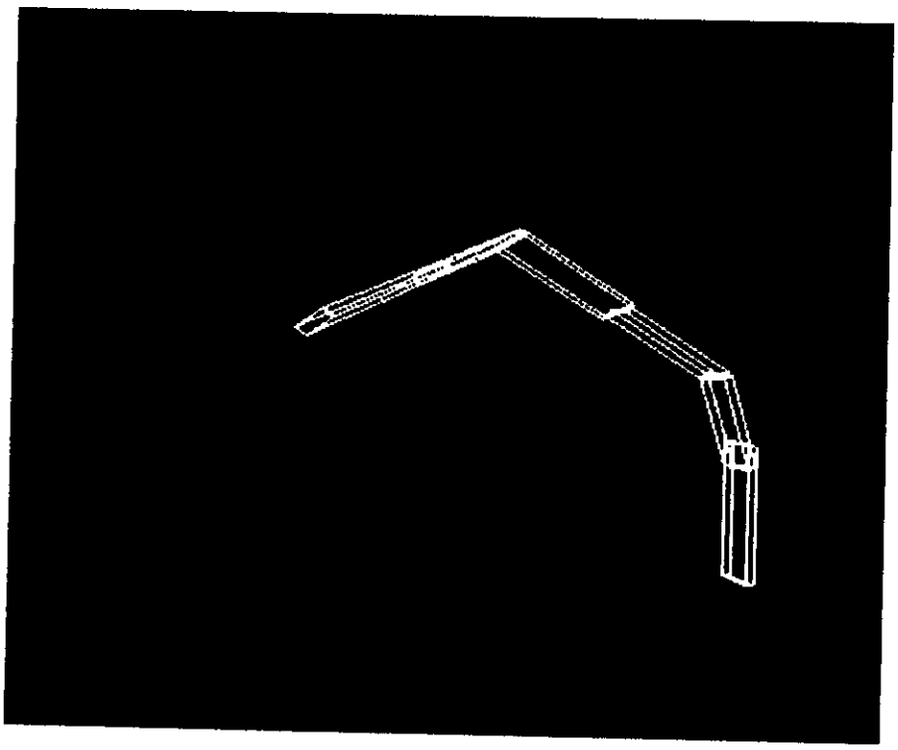


Figure 11 : Animation du bras manipulateur

- **Panneau d'affichage des valeurs des valeurs du capteur d'effort**

Un dernier panneau (figure 12) de l'interface de notre application fourni à l'opérateur en temps réel. Ainsi nous y trouvons l'affichage des valeurs des forces selon les trois axes X, Y et Z et les trois couples autour de ces axes.

CAPTEUR D'EFFO

F(X)	0
F(Y)	0
F(z)	0
T(X)	0
T(Y)	0
T(Z)	0

Figure 12 : Affichage des valeurs du capteur d'effort

CHAPITRE VI

CONCLUSION

CONCLUSION

Le travail présenté dans ce mémoire entre dans le cadre de développement d'outil de commande et de manipulation de robot pour des applications de télésurveillance de sites industriels, lequel est un projet de l'équipe Systèmes Robotisés de Production de la Division Robotique et Productique du Centre de Développement des Technologies Avancées (CDTA).

Le but de notre travail a été de concevoir un système bâtit sur une architecture client/serveur qui permettrait à un utilisateur de :

- contrôler le robot manipulateur mobile (Robuter-ULM du cdta) via un pc hôte. Pour cela il nous fallait d'abord s'assurer que l'échange d'information entre le pc hôte et le robot s'effectuer de façon seine et de manière très rapide pour respecter les contraintes du temps réels du système.

- réaliser des programmes qui permettraient à un opérateur d'accéder aux ressources du robot et ainsi de manipuler ces informations pour qu'il puisse contrôler le robot à partir d'un poste distant.

- rendre l'application facile à utiliser il fallait créer une interface simple et ergonomique peut effectuer toutes les commandes.

Pour cela, nous avons donc développé une application de commande à distance du Robuter-ULM basée sur une architecture client/serveur en utilisant le mécanisme des sockets. Notre application tient compte des deux contraintes majeures :

- un fonctionnement en temps réel (contrainte matérielle),

- un langage de programmation qui peut être interfacé avec l'environnement Syndex qui gère la commande bas niveau.

L'application offre à un opérateur l'accès à toutes les fonctionnalités du robot à partir d'une interface facile à utiliser en respectant la contrainte du temps réel. Ainsi l'utilisateur peut manipuler n'importe quelle ressource du robot de manière simple et évidente et aussi visionner l'état du robot en ayant une vue vertuel du robot à partir des données acquise sur le robot.

Durant notre travail, le problème majeur qui a été rencontré est le conflit entre le fonctionnement de la caméra et celui du bas niveau du robot. En effet les deux processes se discutent la priorité. Après plusieurs tentatives de réduire la priorité de la caméra l'application bas niveau se bloquait toujours après quelques secondes de fonctionnement. Ce problème relève d'ordre matériel qui trouvera sa solution par l'utilisation d'un dispositif de transmission d'image.

Notre travail représente une première étape de la commande à distance du Robuter-ULM du CDTA. De ce fait, nous avons mis l'accent sur la commande en mode manuel. Une suite à notre travail est d'octroyer le robot d'une certaine autonomie sur le plan fonctionnelle ou décisionnelle telle que la planification de trajectoire ou l'évitement d'obstacle. Cette autonomie permet de décharger l'opérateur de certaines tâches et son rôle consistera à superviser le système et intervenir en cas de disfonctionnement.

Bibliographie

[1] Arnaud Leleve, «Télepilotage à Longue Distance d'Engins Mobiles robotisées », thèse Diplôme d'Etude Approfondies en Systèmes Automatiques et Microélectroniques, Option Système Automatique, Juillet 1997, Université de Montpellier II.

[2] Vicente Egea, Jorge Garrido, Roberto Guzman et Ranko Zotovic, «Un Véhicule Autoguide Basé sur Linux », Département des Systèmes et Automatismes, l'Université Polytechnique de Valence.

[3] «Robuter et Bras Ultra Léger : Manuel d'utilisation et de maintenance», Robosoft 2005.

[4] Y. Sorel, «Syndex : Un Logiciel de CAO Niveau Système Pour l'Aide à l'Implantation d'Applications Distribuées Temps Réel Embarquées », INRIA Rocquencourt.

[5] Jean-Damien CARBOU, « Conception d'une architecture pour la commande distance d'un robot d'intervention », thèse Doctorat en Génie Informatique, Automatique et Traitement du signal, Janvier 2004, L'UNIVERSITE DE MONTPELLIER II.

[6] « Réseaux Informatiques »
Les informations se trouvent sur le Site Web: <www.commentcamarche.fr>

[7] Abdelhalim Akka, «Cours de Réseau Informatique», Département d'Informatique, Université de Blida.

[8] Rémi BARLAND et Vincent RAMPAL, «Son Image Réseau : Transmission d'une Séquence Vidéo», Décembre 2002, ESSI (Ecole Supérieur en Science Informatique).

[9] Pascal Nicolas, «Cours de Réseau Maîtrise d'Informatique», Maîtrise d'Informatique, Université d'Angers.

[10] «Architecture Client/Serveur»
Les informations se trouvent sur Site Web <www.centralweb.fr>.

[11] Grady Booch, James Rumbaugh et Ivar Jacobson, «Le Guide de l'Utilisateur UML», Edition EYROLLES.

[12] «UML en Francais»
Les informations se trouvent sur Site Web <www.UML en francais.fr>.

[13] J-C Armici , «Familiarisation avec OpenGL», Janvier 2003.