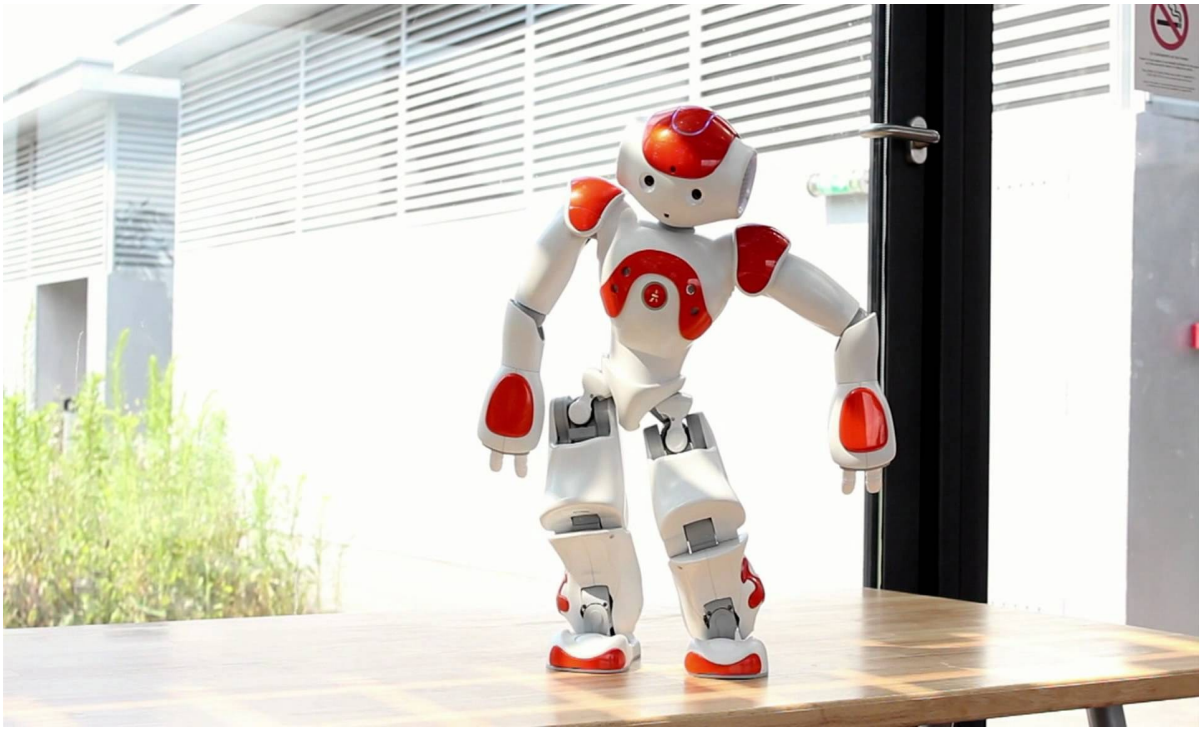
PROJET IMA4

« Synchronisation des mouvements humains à partir d'un robot NAO »



MOIGNER Vincent
THEON Paul
IMA4

Encadrant : Mr Merzouki Rochdi
Année 2012/2013

Remerciements

Nous tenons à remercier Mr MERZOUKI pour son aide et son suivi durant la réalisation du projet ainsi que Mr Scrive concernant sa disponibilité, ainsi que M. Redon qui nous a aidé lors des difficultés d'installation des librairies et programmation python.

Sommaire

Introduction

Dans le cadre de la formation IMA à Polytech'Lille, nous devons effectuer un projet en 4ème année afin de compléter nos compétences.

Nous avons eu la chance de pouvoir réaliser celui-ci sur le robot Nao. Il s'agit d'un robot humanoïde programmable. Ses usages potentiels sont très vastes, on peut l'utiliser par exemple comme robot de compagnie.

Une collaboration existe avec l'école des sciences de l'art de Cambrais afin d'accompagner des danseurs professionnels en mouvements par des robots NAO.

Notre projet consistait à reproduire en temps réel certains mouvements articulaires humains par le robot NAO, ainsi on a dû trouver des solutions techniques pour que le Nao imite instantanément les mouvements d'un danseur.

I ° Présentation du projet

I.1° Cahier des charges

Pour une facilité d'utilisation, les programmes seront créés à partir du système d'exploitation Windows. Mais à cause de problème d'installation de librairies, une partie sera réalisée sous Linux et une partie sous Windows.

Le système devra répondre en temps réel, c'est à dire qu'il faut que le robot NAO puisse suivre quasiment instantanément le danseur. Le choix des capteurs devra être pertinent afin que les données échangées soit facilement transmises, sans trop de pertes, d'erreurs et que la combinaison de capteurs soit non gênante pour l'utilisateur.

Le robot NAO devra respecter le plus possible la chorégraphie du danseur. De plus il sera préférable de communiquer en wi-fi.

I.2° Objectifs personnels

Les projets réalisés durant notre cursus IMA, permettent d'enrichir nos connaissances et nos compétences. Ainsi ce projet permettra tout d'abord d'apprendre à utiliser le robot NAO, puis à programmer en Python et de travailler en autonomie.

De nos jours, les robots font partie intégrante de la vie et le secteur de la robotique est en pleine expansion, c'est pourquoi il est intéressant d'avoir une expérience dans ce domaine.

I.3° Planning

Début du projet, le 28 janvier 2013 à mi-février :

- Découverte et prise en main du robot NAO,

- Prise en main du logiciel Chorégraphe

- Tests sur le NAO

- Choix de capteurs pour acquérir les mouvements d'un bras.

- Constat que chorégraphe n'est pas adapté au temps réel.

- Apprentissage du langage python

Mi-février à fin février :

- Recherche de capteurs pour commencer à tester la fluidité des transferts de données ordinateur -> NAO

- Récupération d'une wiimote

- Récupération des valeurs des capteurs de la wiimote en temps réel sous windows avec le logiciel « GLOVEPIE »

Début mars à fin mars :

- Choix d'utilisation de LINUX pour récupérer les angles de la wiimote

- Programmation d'un serveur/client en python

Début avril à fin du projet :

- Finalisation des programmes

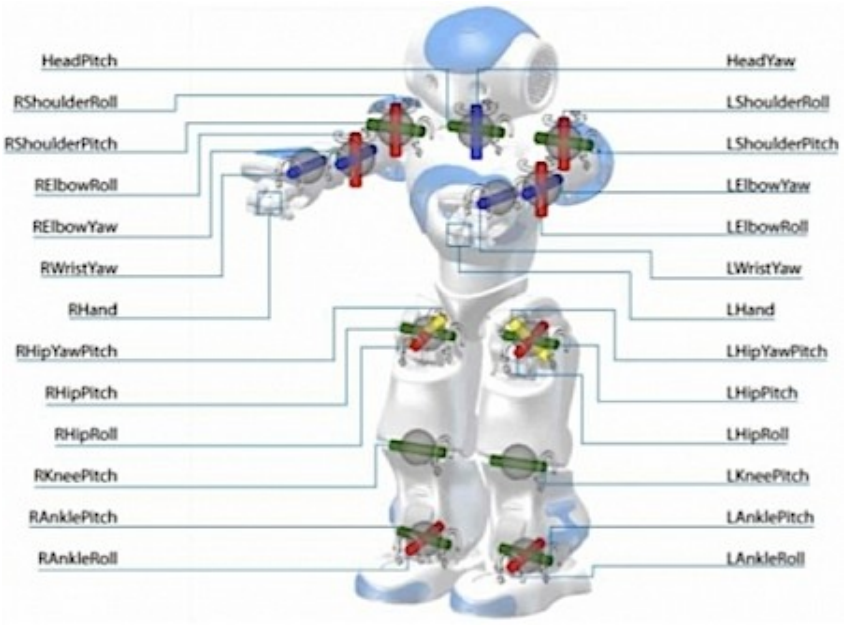
- Amélioration du temps réel

I.4° Présentation de l'environnement

On avait à disposition pour notre projet, un robot NAO, une borne wifi, une manette wii, un adaptateur bluetooth et les logiciels de programmation tel que Choregraphe.

I.4.1° Le robot Nao

Le robot NAO est un robot humanoïde autonome et programmable de 58cm, à 25 degrés de libertés, dont les principaux éléments sont les moteurs et actionneurs électriques, mis en évidence ci-dessous.



Représentation des moteurs du robot

Pour l'instant, il est utilisé principalement par des laboratoires de recherche et pour l'enseignement. Il a été présenté au public fin 2006 par l'entreprise française Aldebaran robotics.

Le NAO se connecte et communique par Ethernet ou en Wi-fi. Il est doté d'un impressionnant réseau de capteurs : deux caméras, quatre microphones, un télémètre sonar, deux émetteurs et récepteurs infrarouge, une carte inertielle, neuf capteurs tactiles et huit capteurs de pression.

Enfin, sa batterie de 27,6 watt-heure confère au NAO environ une heure et demie d'autonomie, en usage normal.

I.4.2° Langages de programmation pour le robot

Le robot NAO est programmable en C++, Python, Java, MATLAB, Urbi, C, et Net.

Nous avons choisi de programmer le NAO en python car c'est le langage utilisé en règle générale pour faire du temps réel et les librairies sont téléchargeables facilement depuis le site d'Aldebaran robotics. De plus, il est indiqué expressément que python est le langage le plus simple pour programmer le robot NAO. C'est pourquoi, nous avons dû apprendre ce langage que nous n'avions jamais vu auparavant.

I.4.3° Les logiciels utilisés pour la programmation du NAO

Chorégraphe est le logiciel le plus utilisé pour programmer le robot NAO. Cependant, il ne convenait pas du tout dans notre projet car il permet de réaliser des programmes séquentiels (pour le robot), or nous devions faire un programme répondant

en temps réel à des événements, et non des séquences enregistrées. Ainsi, pour exécuter notre code python, on a utilisé l'extension « python2.7 ».

I.4.4° Le python

Le langage python est un langage plutôt facile à apprendre, à comprendre et à écrire. Il est portable car il fonctionne sous de nombreux systèmes d'exploitation. Il est adapté aussi bien pour des scripts, des petits ou gros projets. De plus il est doté d'une façade objet bien conçue et puissante.

Compilation du python :

Pour compiler un programme Python sous Windows, il faut ouvrir l'éditeur IDLE (après avoir installé Python2.7), écrire son programme puis simplement l'exécuter (dans l'onglet Run → Run Module).

Sous Linux, il suffit de se placer dans le bon répertoire sur le terminal et d'entrer la commande suivante : ***$ python programme.py*** .

I.4.5° Structure logiciel du NAO

Le principal logiciel qui fonctionne sur le robot et le contrôle, est «NAOQqi» Il s'agit de l'interface de programmation, qui va permettre de traduire le code python (ou autres) pour le robot.

L'exécutable « NAOqi » lors du démarrage, charge et définit toutes les librairies nécessaires à partir du fichier « autoload.ini » (voici ci-dessous la structure de NAOqi).

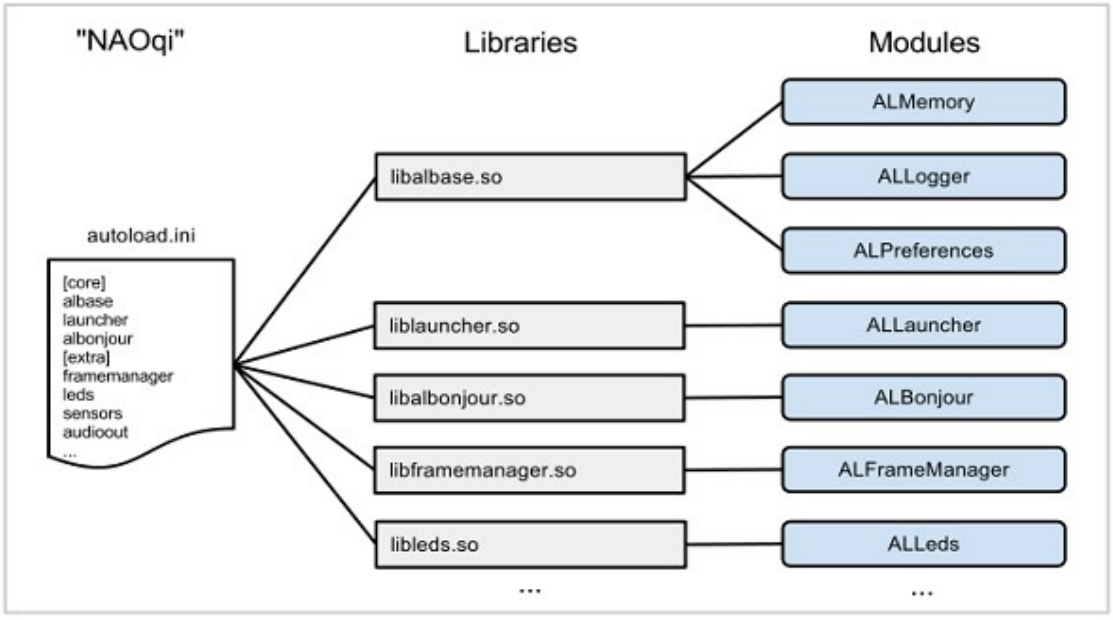


Schéma structurel de NAOqi

NAOqi réalise également l'accès réseau.

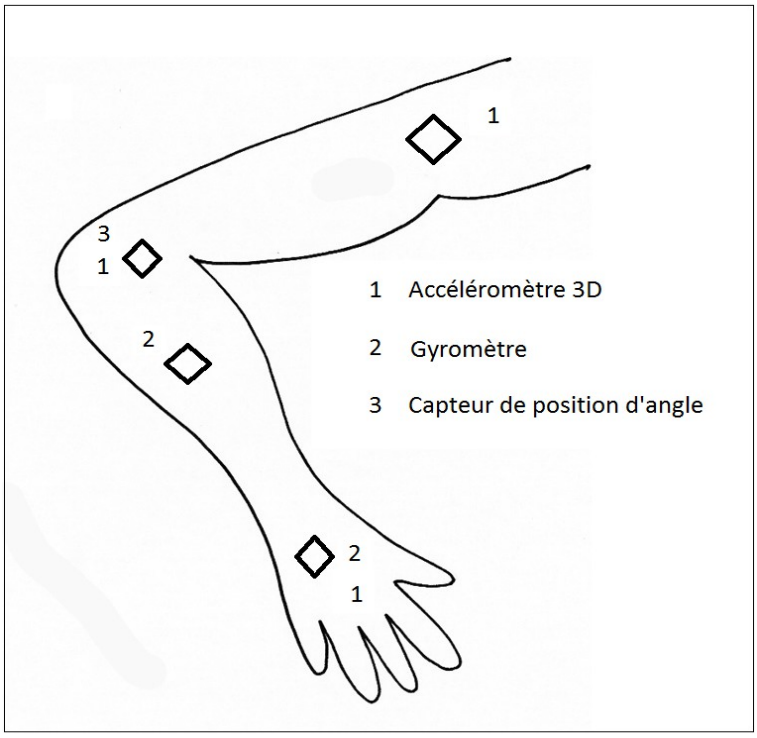
Connexion au robot NAO :

Le code python suivant est un exemple permettant juste au robot de dire « Hello, word ! ». Il faut importer le module « ALproxy » puis connaître l'adresse IP du robot (en appuyant sur le bouton centrale du NAO).

```
from naoqi import ALProxy
tts = ALProxy("ALTextToSpeech", "<IP of your robot>", 9559)
tts.say("Hello, world!")
```

II ° Travaux réalisés

Nous avons dû imaginer une combinaison de capteurs permettant de capter les mouvements des différentes articulations du danseur.



Représentation des capteurs sur un bras

Une fois celle-ci validée par M. Merzouki, nous avons commencé à travailler sur la mise en œuvre du transfert de données.

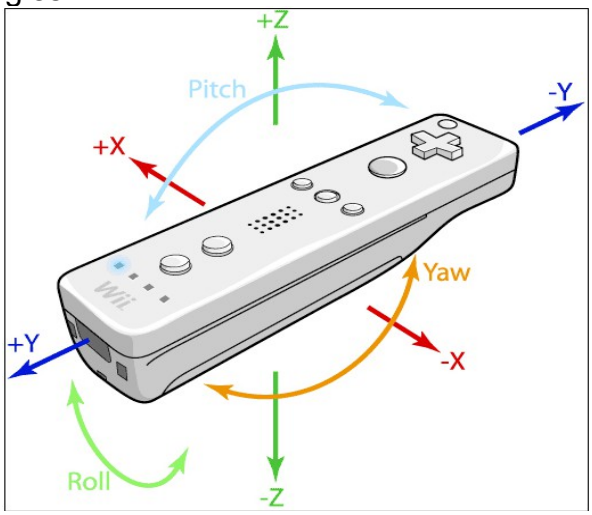
Dans un premier temps, nous avons choisi de travailler sur un seul capteur pour tester la transmission des données en temps réel.

II.1 ° Utilisation de la wiimote

A Polytech Lille, des études ont été menés sur le contrôle des robots KUKA à partir d'une manette wii. Ainsi on a utilisé cet outil pour contrôler un moteur du NAO et en particulier les articulations de la tête. Une fois que le test sera validé, il faudra réaliser la même chose pour toutes les autres articulations.

II.1.1 ° Présentation de la manette wii

Nous avons choisi la tête car celle-ci a deux degrés de liberté contrôlés par les moteurs « Headyaw » et « Headpitch ». Ces articulations seront contrôlées respectivement par les angles « Yaw » et « Roll » de la wiimote, voici ci-après une représentation de ces angles.

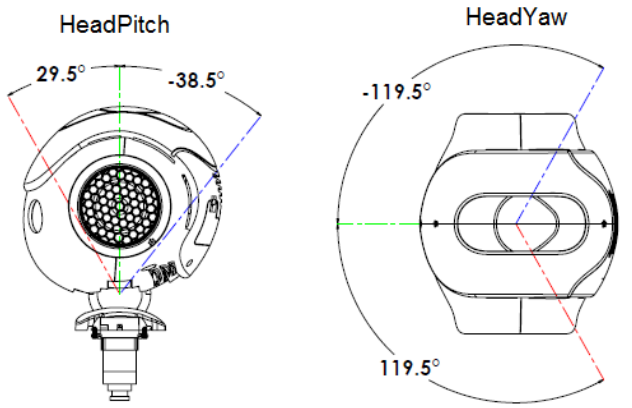


Représentation des angles sur la wiimote

II.1.2 ° Limite des articulations du robot NAO

Le robot a une certaine limite vis à vis des angles d'inclinaison des articulations. Par exemple, pour l'articulation « HeadPitch », l'angle d'inclinaison est compris entre 29,5° et -38,5° et l'articulation « HeadYaw », -119,5° et 119,5°.

Il faut prendre en considération cette limite dans le programme afin de ne pas détériorer le NAO, il en va de même concernant les autres articulations du robot.



Joint name	Motion	Range (degrees)	Range (radians)
HeadYaw	Head joint twist (Z)	-119.5 to 119.5	-2.0857 to 2.0857
HeadPitch	Head joint front and back (Y)	-38.5 to 29.5	-0.6720 to 0.5149

Limites des articulations

II.1.3 ° Calcul des angles « pitch » et « roll »

La wiimote comporte 3 accéléromètres, et la librairie utilisée « Cwiid » ne permet pas de récupérer directement les valeurs des angles gyroscopiques. Nous avons donc du calculer à partir des accéléromètres les angles « pitch » et « roll ». Pour cela on a analysé les valeurs des accéléromètres en faisant varier les angles d'inclinaison de -90 à 90 degrés et on a ainsi déterminé les équations suivantes.

Calcul des accélérations suivant les axes x,y et z :

x=mesg[1][cwiid.X]
y=mesg[1][cwiid.Y]
z=mesg[1][cwiid.Z]

Résolution des angles « pitch » et « roll » :

pitch=(x-123)*3.6
roll=(123-y)*3.6

II.1.4 ° La librairie CWIID

La manette wiimote communique en Bluetooth avec l'ordinateur. Elle nous a donc permis de commencer à travailler sur le transfert de données en temps réel.

Dans un premier temps on a utilisé le logiciel GLOVEPIE fonctionnant sous Windows afin de recevoir les angles « pitch » et « roll ». Du fait de la complexité du logiciel et de la récupération des angles, on a décidé sous l'accord de Monsieur Merzouki de basculer sous Linux. En effet, il existe une librairie qui est la plus utilisée et développée

sur internet permettant de récupérer les angles en question, il s'agit de la librairie « Cwiid ».

II.2 ° Gestion des programmes et de l'environnement

II.2.1 ° La Communication serveur client

Nous avons choisi de faire une communication serveur client entre deux ordinateurs, un sous Windows et l'autre sous Linux. En effet les sockets python sont relativement simples à mettre en place et elles nous permettent de faire transiter les données via le réseau d'un ordinateur « acquisition des données capteur » à un ordinateur « commande du robot NAO ». Nous expliqueront plus tard pourquoi l'utilisation de deux ordinateurs a été nécessaire (voir partie 3).

Configuration serveur client :

Sur l'ordinateur Windows allez dans :

→ Panneau de configuration\Réseau et Internet\Connexions réseau

→ modifier les paramètre de la carte

→ Propriété de connexion au réseau local → ipv4(propriétés) → adresse ip : 192 168 0 100 et MASQUE : 255 255 255 0

Sur l'ordinateur Linux :

A la ligne de commande on va forcer une adresse IP:

#ifdown eth0
#ifconfig eth0 192 168 0 101

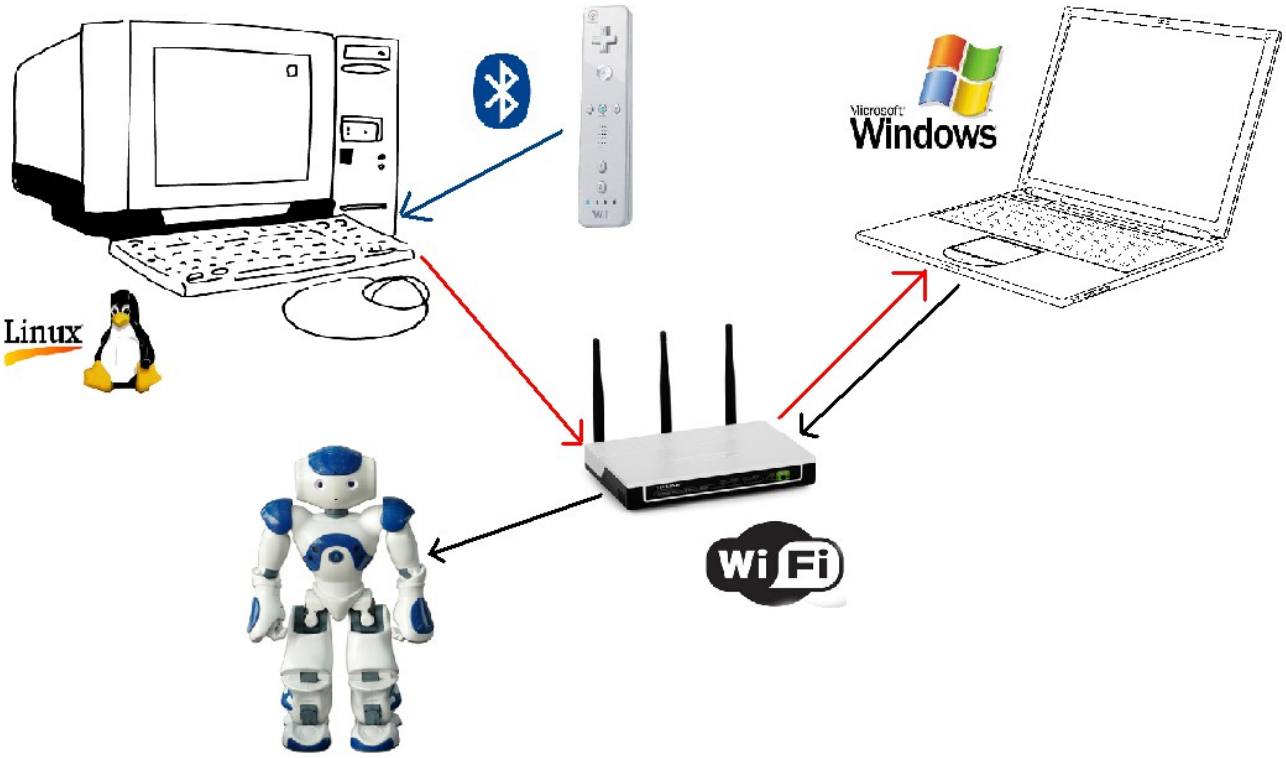
On vérifie la communication en faisant un test de ping.

Attention, le test de ping ne peut être fait que de Windows vers Linux, car Windows ne répond pas au ping.

Attention de toujours exécuter le programme serveur avant de lancer le programme client (voir les programmes serveur/client python en annexe1).

II.2.1 ° Structure du réseau

Voici ci-dessous la structure du réseau, l'ordinateur sous Linux permet de récupérer les angles « pitch » et « roll ». Ensuite ils sont envoyés à l'ordinateur sous Windows par le biais de la borne Wifi et pour finir, ces données sont filtrés et envoyer au NAO.



Structure du réseau

II.2.1 ° Fonctionnement des programmes

Un premier programme python permet principalement de récupérer les valeurs des angles pitch et roll de la wiimote puis il envoie ces angles via la communication serveur/client. Ce programme correspond au client.

Un deuxième programme python permet de récupérer la valeur d'un angle (il s'agit du serveur) et de l'envoyer au robot NAO par wi-fi, tout en connaissant son adresse ip (annexe 3).

Le temps réel se gère principalement en réglant la valeur du temps d'exécution de l'ordre au moteur de l'articulation du NAO, car les valeurs arrivent en nombre trop important, il faut donc ralentir le traitement pour ne pas submerger la bande passante, auquel cas le NAO répond de moins en moins vite aux consignes. Il faut aussi gérer le transfert des angles de manière à garder un temps de réponse constant, concordant avec le temps d'exécution du NAO et le plus faible possible.

III ° Difficultés rencontrées et solution choisie

Nous avons perdu beaucoup de temps en cherchant à recueillir les valeurs de la wiimote sur Windows car aucune librairie python ne fonctionnait sous Windows. Nous avons finalement demandé et obtenu l'autorisation de poursuivre le projet sous Linux.

Sous Linux, nous avons pu travailler avec la wiimote sans aucun problème. Mais nous avons ensuite eu des difficultés avec la librairie NAOqi. En effet après avoir correctement installé la version Linux 64bit sous Linux, elle ne semblait pas vouloir fonctionner avec notre programme créé sous Windows.

Après avoir cherché des solutions fonctionnant exclusivement sous Linux, la solution que nous avons choisi est d'utiliser deux ordinateurs. L'un sous Windows et l'autre sur Linux.

L'ordinateur sous Linux récupère les valeurs de la wiimote et les envoie à l'ordinateur sous Windows.

L'ordinateur sous Windows envoie les données au robot NAO.

Pour la communication entre les deux pc, on a choisi de faire des sockets python (voir annexe 1) car nos programmes étaient déjà en python. En effet l'ordinateur sous Windows sera le client, qui emmétra les données vers le serveur, l'ordinateur Linux.

Conclusion

En conclusion, après ces 3 mois de travail, nous sommes satisfait d'avoir réussi à obtenir une solution quasi-fonctionnelle, respectant presque le cahier des charges pour l'aspect temps-réel.

Il ne restera plus qu'à améliorer l'aspect précision, puis généraliser nos programmes à tous les moteurs du Nao et à mettre en application une combinaison de capteurs en se basant sur le travail effectué sur la wiimote.

Ce projet mêlant art et technologie nous a permis de travailler sur l'un des fleurons de la technologie française, tout en ayant l'opportunité d'utiliser toutes les compétences que nous avons acquises à Polytech'Lille en IMA, telles que le réseau, la programmation, sur un système autonome.

annexe 1 : programmes serveur / client

Le serveur :

```
import socket
Sock = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
Host = '192.168.0.100' # l'ip locale de l'ordinateur
Port = 234     # choix d'un port au hasard


Sock.bind((Host,Port))

# On est a l'ecoute d'une seule et unique connexion :
Sock.listen(1)


client, adresse = Sock.accept() # accepter les connexions de l'extérieur
print "L'adresse",adresse,"vient de se connecter au serveur !"
while 1:
        RequeteDuClient = client.recv(4) # on reçoit 4 caractères max
        if not RequeteDuClient: # si on ne reçoit plus rien
                break  # on sort de la boucle
        print RequeteDuClient     # affiche les données envoyées
```

Le client :

```
import socket
Sock = socket.socket(socket.AF_INET,socket.SOCK_STREAM) # on crée notre socket


Host = '192.168.0.100'
Port = 234

# on se connecte sur le serveur avec les informations ci-dessus
# assurez-vous d'avoir mis en marche le serveur !
Sock.connect((Host,Port))

# On est connecte, on fait une boucle infinie d'inputs pour l'envoi des messages :
while 1:
        msg = raw_input('>> ')  # on rentre des données au terminal pour tester la
socket
        Sock.send(msg) # on envoie ces données
```

annexe 2 : programme wiimote (client)

```
#!/usr/bin/python
import socket
import cwiid
import sys
import time

# on cree notre socket
Sock = socket.socket(socket.AF_INET,socket.SOCK_STREAM)

# definition des informations de la socket:
Host = '192.168.0.100'       #ip de l'ordinateur sous windows
Port = 234
# mettre en marche le serveur avant de lancer le client !

Sock.connect((Host,Port))

def main():
        rpt_mode = 0
        mesg = False

        #connexion de la wiimote
        print 'allumez la wiimote (appuyez 1+2)...'
        if len(sys.argv) > 1:
                wiimote = cwiid.Wiimote(sys.argv[1])
        else:
                wiimote = cwiid.Wiimote()

        wiimote.mesg_callback = callback

        print 'appuyez sur "a" pour lancer le programme'

        while(1):


                c = sys.stdin.read(1)
                if c == 'a':
                        mesg = not mesg
                        if mesg:
                                wiimote.enable(cwiid.FLAG_MESG_IFC);
                        else:
                                wiimote.disable(cwiid.FLAG_MESG_IFC);

                        rpt_mode ^= cwiid.RPT_ACC
                        wiimote.rpt_mode = rpt_mode

        wiimote.close()

def callback(mesg_list, dummy):
```

```
        for mesg in mesg_list:

#envoi des donnees maintenant vers l'ordinateur windows :

                        Sock.send(pitch)
                        Sock.send('\n')

main()
                if mesg[0] == cwiid.MESG_ACC:
                        x=mesg[1][cwiid.X]
                        y=mesg[1][cwiid.Y]
                        z=mesg[1][cwiid.Z]

                        pitch=(x-123)*3.6
                        roll=(123-y)*3.6
                        print 'pitch, roll: pitch=%d roll=%d' % (pitch,roll)

# on veut envoyer des entiers mais la socket ne peut transporter que des strings

                        pitch=str(int(pitch))

#envoi des donnees maintenant vers l'ordinateur windows :

                        Sock.send(pitch)
                        Sock.send('\n')

main()
```

annexe 3 : programme Nao (serveur)

```
import sys
import math
import time
from naoqi import ALProxy
from time     import sleep
import socket


def main(robotIP,capteur1):                    #cette fonction commande le Nao
    try:
        proxy = ALProxy("ALMotion", robotIP, 9559)
    except Exception, e:
        print "Could not create proxy to ALMotion"
        print "Error was: ", e

    proxy.setStiffnesses("Head", 1.0)

    # ici on bouge le moteur de la nuque du Nao de l'angle capteur1 pendant une durée de
0,5 secondes
    names  = "HeadYaw"
    angles = (capteur1*(math.pi))/180
    print angles,"\a"
    times  = 0.5
    isAbsolute = True
    proxy.post.angleInterpolation(names, angles, times, isAbsolute)



if __name__ == "__main__":
    robotIp = "192.168.0.9"

    if len(sys.argv) <= 1:
        print "Usage python motion_taskManagement1.py robotIP (optional default:
127.0.0.1)"
    else:
        robotIp = sys.argv[1]


Sock = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
Host = '192.168.0.100' # l'ip locale de l'ordinateur
Port = 234         # choix d'un port

Sock.bind((Host,Port))

# On est a l'ecoute d'une seule et unique connexion :
Sock.listen(1)
```

```
# Le script se stoppe ici jusqu'à ce qu'il y ait connexion :
client, adresse = Sock.accept() # accepte les connexions de l'extérieur
print "L'adresse",adresse,"vient de se connecter au serveur !"

u=0
while 1:


    RequeteDuClient = client.recv(3) # on recoit 3 caracteres grand max (de -90 à +90)
    if not RequeteDuClient: # si on ne recoit plus rien
        break
    try:
        capteur1 = int(RequeteDuClient)
    except ValueError:
       print('Entrée incorrecte')

#ici on gère le temps réel en évitant de traiter trop de valeurs acquises

    if(-100 < capteur1 < -80): capteur1=-90
    if(-80 < capteur1 < -60): capteur1=-70
    if(-60 < capteur1 < -40): capteur1=-50
    if(-40 < capteur1 < -20): capteur1=-30
    if(-20 < capteur1 < 0): capteur1=-10
    if(0 < capteur1 < 20): capteur1=10
    if(20 < capteur1 < 40): capteur1=30
    if(40 < capteur1 < 60): capteur1=50
    if(60 < capteur1 < 80): capteur1=70
    if(80 < capteur1 < 100): capteur1=90


    if (capteur1!=u):
       print capteur1,"\a"
       print '\n'
       u=capteur1
       main(robotIp,capteur1)
       u=capteur1
```

Annexe 4 : Notice d'utilisation du projet :

Sur l'ordinateur Windows, installer python2.7, puis configurer l'adresse IP.

Sur l'ordinateur Linux, installer la librairie CWIID : sudo apt-get install cwiid

Lier les ordinateur sur un réseau local avec la borne wi-fi.

Démarrer le nao est vérifier qu'il est connecté au réseau wi-fi.

Lancer le programme serveur sur l'ordinateur Linux.

Lancer le programme client sur l'ordinateur Windows. Suivre les instructions (allumer wiimote...)