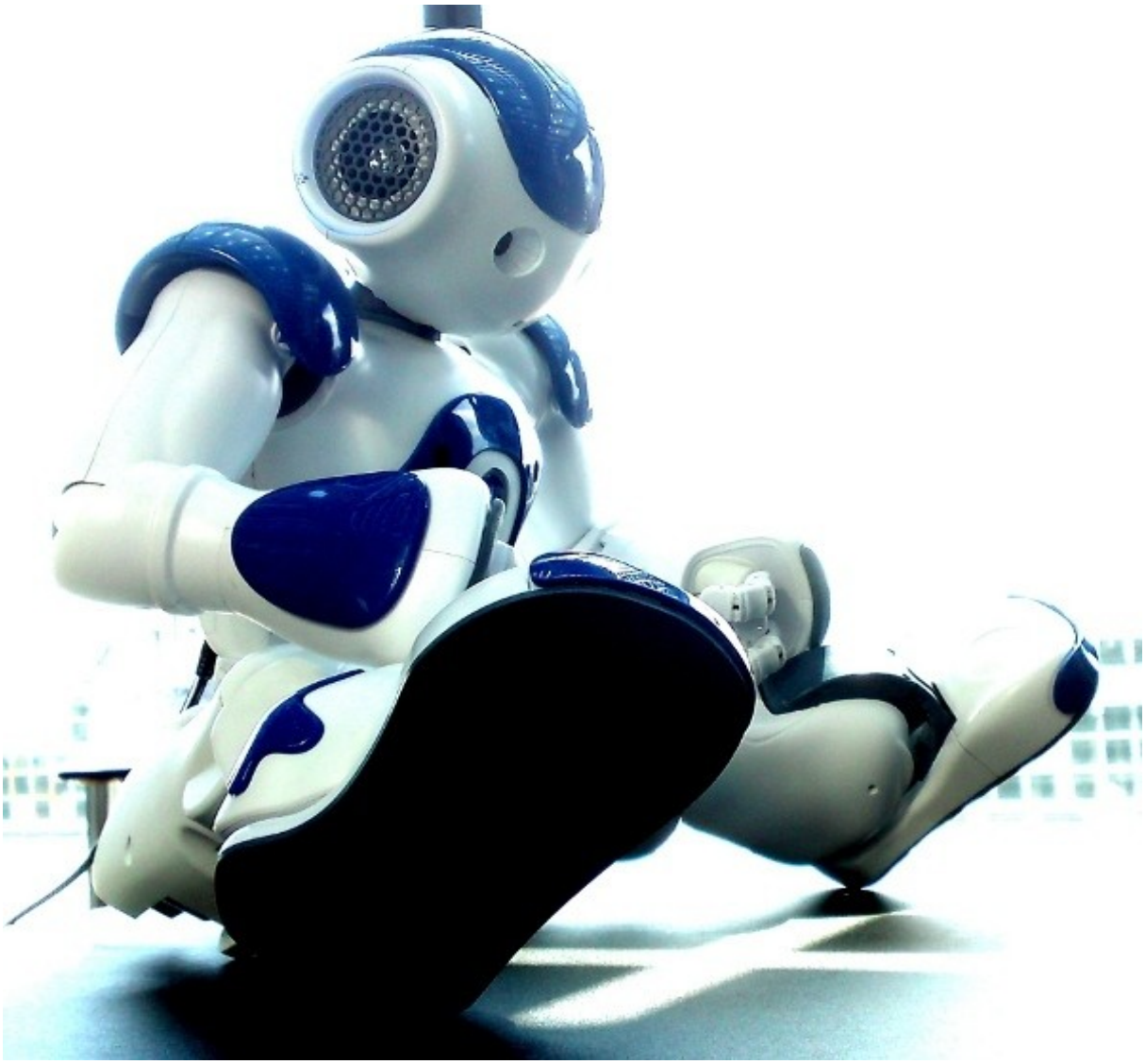
HENRY Charles
WALLET Xavier

Encadrant:
MR MERZOUKI Rochdi

PROJET NAO

Objectif du projet:
Faire coopérer 2 robots humanoïdes NAO afin de réaliser un groupe de musique et une chorégraphie, le tout de manière synchronisée.



Remerciements

Nous tenons à remercier MR MERZOUKI Rochdi pour son suivi, son aide et sa présence tout au long de ce projet, MR SCRIVE Olivier pour sa disponibilité et la confiance qu'il nous a donnés, MR CONRARD Blaise pour ses explications ainsi que MR REDON Xavier et MR VANTROYS Thomas , responsables des stages IMA4.

Table des matières

Introduction:

NAO est un robot humanoïde autonome, programmable développé par la société française Aldebaran Robotics. Cette société française a été créé en 2005 par Bruno MAISONNIER.

Elle développe actuellement plusieurs modèles de robots, dont l'humanoïde autonome et programmable NAO (mis sur le marché en 2010).

Les caractéristiques du robot NAO sont les suivantes:

Caractéristiques techniques de NAO	
Hauteur	58 cm
Poids	4,8 kg
Autonomie	90 min
Degrés de liberté	26
Processeur	Intel ATOM 1,6GHz
Système d'exploitation intégré	Linux
Systèmes d'exploitations compatibles	Windows, Mac OS, Linux
Langages de programmation	C++, Python, JAVA, MATLAB, Urbi, C, Net
Connectivité	internet, Wi-Fi

Le robot NAO possède 26 degrés de libertés pour ses déplacements, afin de reproduire au mieux les mouvements d'un humain. Il est également équipé de différents capteurs et émetteurs:
-capteurs ultrason
-capteurs de pressions résistifs
-un système multimédia (microphones, haut parleurs, caméras HD.
-capteurs tactiles
-LED infrarouges (émetteur et récepteur)
-connexion WIFI.

Avec tout son équipement, La programmation d'un NAO pour effectuer une tache est quasi sans limites (envoyer un mail, jouer au football, ramener un objet donné...)
Polytech'lille dispose de trois robots NAOs, la programmation du NAO est possible grâce au logiciel de programmation Choregraphe fourni par la société lors de l'achat d'un NAO.
Notre projet consiste à faire coopérer deux robots humanoïdes NAO afin de réaliser un groupe de musique et une chorégraphie.

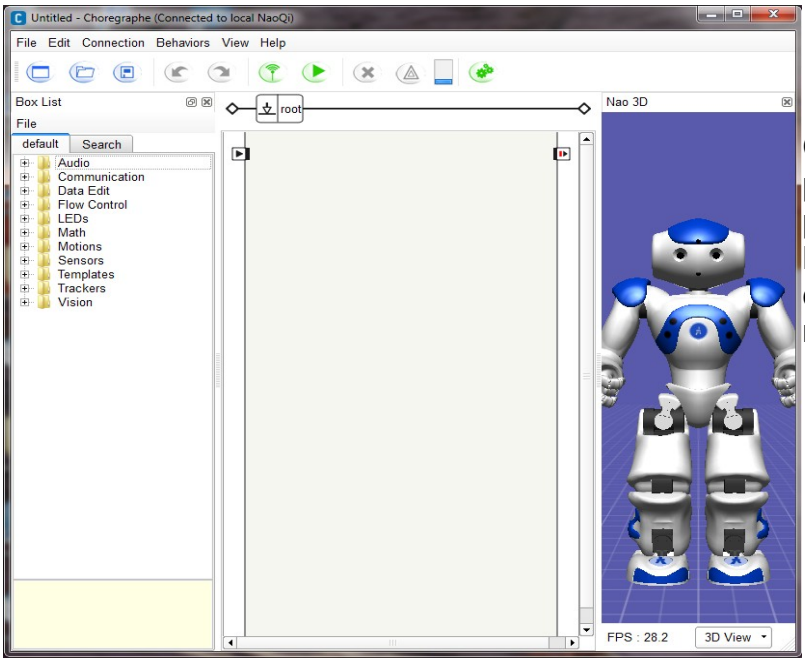
Nous allons vous présenter les moyens mis en œuvre pour répondre au cahier des charges...

Contexte du projet

Notre formation IMA nous sensibilise à l'étude de systèmes complexes dans le but de répondre à un cahier des charges fourni par une entreprise. La robotique est un secteur d'activité en pleine expansion, le NAO est un robot humanoïde qui dans un futur proche pourra être amené à accompagner dans la vie de tous les jours différentes personnes.
En accord avec notre formation, il est donc intéressant d'étudier cette nouvelle technologie afin de répondre à un cahier des charges imposé...

I - Prise en en main de la programmation d'un NAO

a) présentation du logiciel



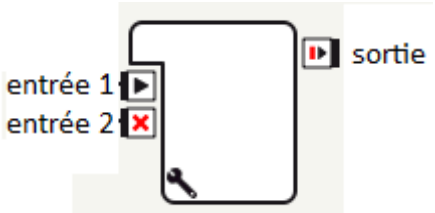
Choregraphe est le logiciel de programmation de NAO fourni par son constructeur, Aldebaran. L'aide de ce logiciel est très complète et nous permet une rapide prise en main.

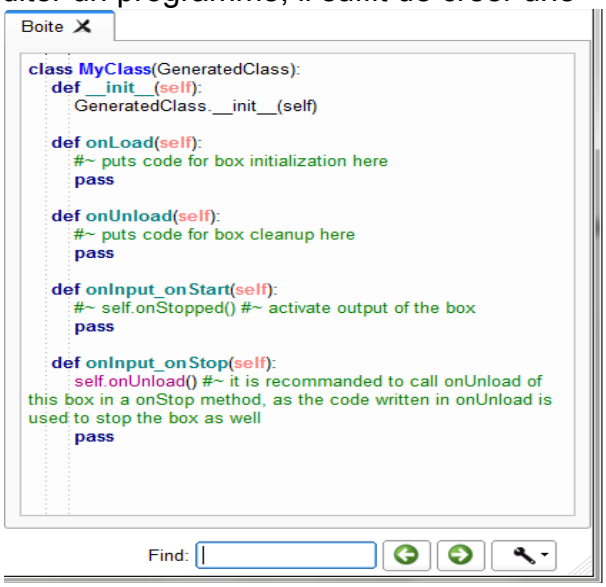
Pour programmer un NAO, il « suffit » de relier, en série ou en parallèle, différentes boites de commande, chaque boite renferme une action à exécuter par le NAO.

On distingue deux types de boite de commande:
-Boites de commande renfermant un langage python
-Boîtes de commande de types « Animation » permettant d'enregistrer la position dans le temps des différents moteurs (et donc permettre au NAO de reproduire un mouvement).

b) la programmation d'une boite de commande en python

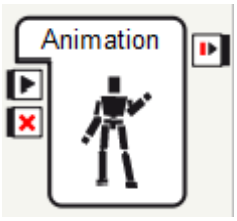
Nous disposons d'une bibliothèque contenant les différentes fonctions (en python) utilisables pour programmer le robot. Pour éditer un programme, il suffit de créer une





nouvelle boite commande:
Une boite de commande est composée de deux entrées et d'une sortie, lorsqu'une de ses entrées est activée par un signal extérieur, le programme associé à cette entrée est exécuté. On retrouve en effet dans notre code python un espace de programmation dédié à chaque entrée
Def onInput_onStart(self): les lignes de codes associés sont exécutées si un signal d'entrée est détecté sur l'entrée 1.
Def onInput_onStop(self): les lignes de codes sont exécutées si un signal d'entrée est détecté sur l'entrée 2.
L'activation de la sortie est quant à elle générée une fois le programme terminé.
Afin de comprendre et surtout de mieux maîtriser la programmation d'un NAO sous python, nous nous nous sommes exercés sur quelques exemples (Allumer les LEDs du NAO d'une certaine couleur, utiliser les détecteurs tactiles, émettre un son, ….).

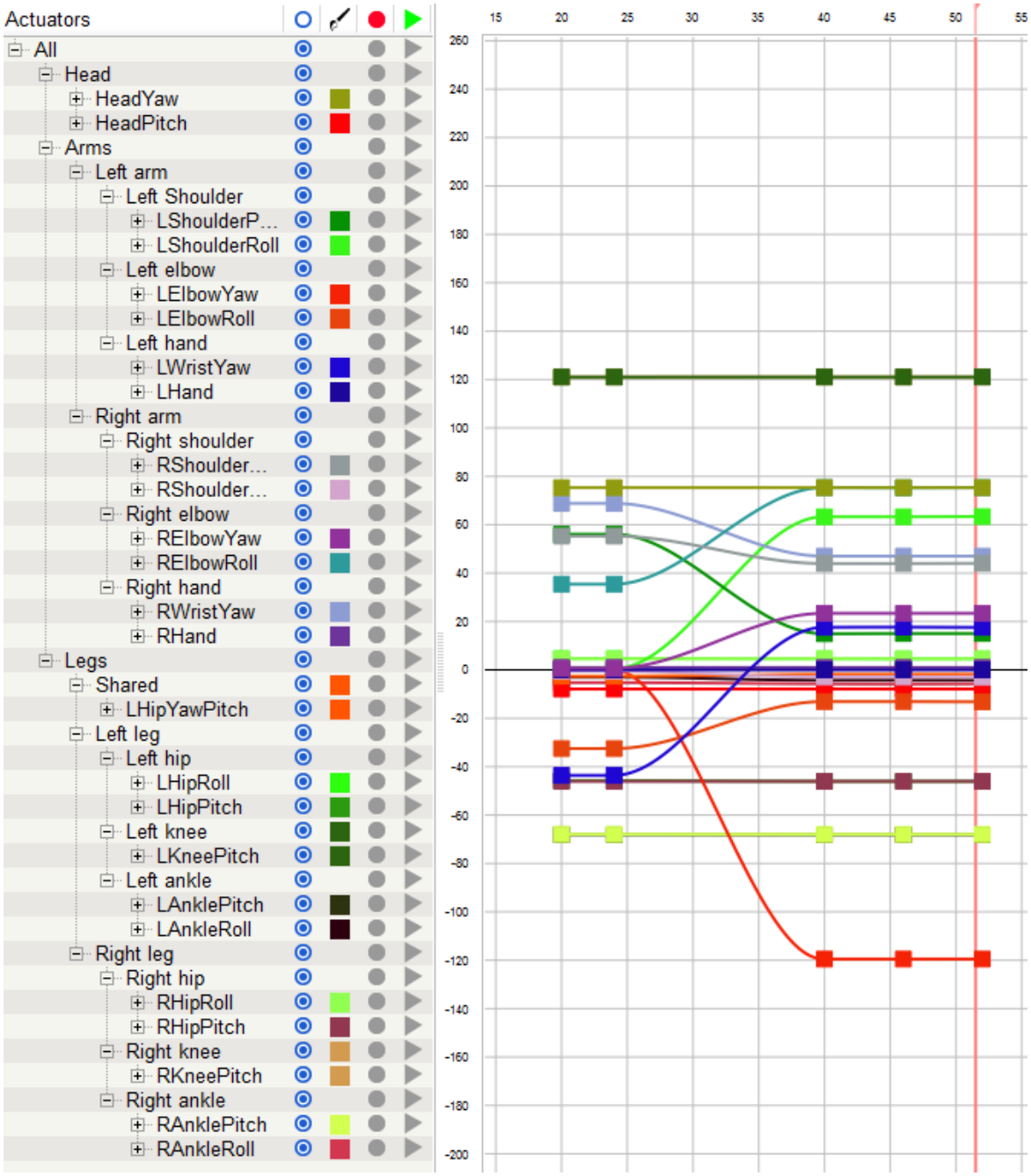
c) L'enregistrement d'un mouvement



Il existe un autre type de boite de commande, les boites de types « animation ».Ces boites de commande permettent d'enregistrer la position de chaque moteur en fonction du temps.
L'exécution de cette boite permet ensuite au NAO de reproduire le mouvement enregistré.
Le bloc Animation est très utile pour des mouvements « statiques », c'est à dire des mouvements où le robot n'entreprend pas de réel déplacement. Nous utiliserons donc ce système pour apprendre à notre NAO différents mouvements.

Comme nous l'avons dit précédemment, NAO dispose de 26 degrés de liberté. Pour un mouvement enregistré, nous disposons de 26 courbes, chaque courbe traduit le déplacement dans le temps d'un moteur, sa rotation.
Il est possible ensuite de retravailler manuellement chaque courbe (points par points) afin d'optimiser notre mouvement dans l'espace et de le rendre le plus naturel possible (rapidité, points intermédiaires, ...) .
Le contrôle de chaque moteur est possible, nous disposons de la liste et de la spécificité de chaque moteur:

Moteur	Degrés de liberté	Nombre
Head	2	1
shoulder	2	2
elbow	2	2
hand	2	2
shared	1	2
hip	2	2
knee	1	2
ankle	2	2



exemple de l'enregistrement d'un mouvement

Nous avons vu dans cette partie les différents outils dont nous disposons afin de répondre à notre cahier des charges.
L'association de bloc animation pour les mouvements des robots ainsi que de bloc de commande en python pour la gestion des différentes fonctions seront utilisées
Une fois le logiciel de programmation maîtrisé et notre cahier des charges défini, nous pouvons nous pencher sur les différentes problématiques de notre projet.

II - La communication entre les NAOs

a) Les différents types de communications

La communication entre les NAO, c'est à dire l'échange d'informations, est la clé de notre projet.

Il existe 3 types de communications entre NAO :
-L'envoi d'information par infrarouge (chaque robot dispose d'un émetteur et d'un récepteur infrarouge dans son œil droit)
-L'échange d'informations par WIFI (les NAOs étant connectés sur un même routeur)
-L'échange d'informations par bluetooth.

La documentation technique du NAO nous donne différentes informations et exemples d'utilisations. Nous avons choisi d'utiliser une communication infrarouge. Malheureusement nos robots, n'étaient pas équipés d'une carte bluetooth et nous manquions d'informations (aucune) concernant le WIFI. Nous avons demandé à la société Aldebaran un exemple d'utilisation du WIFI mais leur réponse a été trop tardive pour utiliser cette méthode de communication.

Cette étude nous a permis d'analyser les différentes possibilités pour échanger des informations entre deux NAO, la documentation technique est plus ou moins complète sur l'échange Infrarouge, nous allons nous intéresser a cette dernière méthode …

b) La communication Infrarouge

Nous avons vu que l'échange d'information par infrarouge était possible entre deux NAO. La programmation associée à cette échange d'informations se fera en langage python (comme nous l'avons vu précédemment, le code étant renfermé dans une boite de commande).
Pour cela nous disposons d'une bibliothèque avec les différentes fonctions en python propre au NAO.

Nous allons donc utiliser ces différentes fonctions afin de résoudre notre première problématique.

Nous échangeons des informations entre les robots afin de les synchroniser (autrement dit de les faire coopérer) afin qu'ils débutent ensemble ,par exemple, un même mouvement ou qu'ils s'envoient différentes informations durant l'exécution d'un programme pour créer un échange et donc une possibilité de coopération par la suite.

Nous allons maintenant nous intéresser à la réception et l'émission d'un octet en infrarouge.

Voici notre boite de commande (avec le code python associé) pour la réception d'un octet :

La réception d'un octet par infrarouge:

code python:

```
class MyClass(GeneratedClass):
    def __init__(self):
        GeneratedClass.__init__(self)
        self.hw = ALProxy("ALInfrared")
        #self.hw.subscribetomicroevent

    def onLoad(self):
        self.blsRunning = False

    def onUnload(self):
        self.blsRunning = False

    def onInput_onStart(self):
        self.blsRunning = True
        self.hw.initReception(-1)
        #~ self.onStopped() #~ activate output of the box
        pass

    def onInput_onStop(self):
        if( self.blsRunning ):
            self.onUnload() #~ it is recommanded to call onUnload of this box in a
onStop method, as the code written in onUnload is used to stop the box as well
            self.onStopped()
```

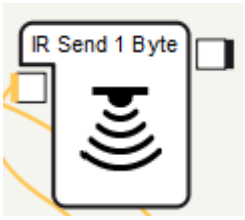
Explications du programme :

On retrouve « le squelette » de base d'un programme python sous NAO.
Pour activer le récepteur infrarouge, il suffit de l'initialiser à -1. Pour cela, nous utilisons la fonction initReception associée au récepteur infrarouge .
La fonction ALLProxy( ) nous permet de sélectionner le récepteur infrarouge (AllInfrared étant le mot clef de la bibliothèque).

Au final notre boite de commande émet un signal de sortie lorsque elle reçoit un octet par infrarouge.

Voici notre boite de commande (avec le code python associé) pour l'émission d'un octet :

L'émission d'un octet:



code python:

```
class MyClass(GeneratedClass):
    def __init__(self):
        GeneratedClass.__init__(self)
        self.hw = ALProxy("ALInfrared")

    def onLoad(self):
        #~ puts code for box initialization here
        pass

    def onUnload(self):
        #~ puts code for box cleanup here
        pass

    def onInput_onData(self, data):
        self.hw.send8(data)
        self.onSent()
        pass

    def onInput_onStop(self):
        self.onUnload() #~ it is recommanded to call onUnload of this box in a onStop
method, as the code written in onUnload is used to stop the box as well
        pass
```

De la même manière que pour la réception, nous utilisons les fonctions prédéfinies dans la bibliothèque.

La fonction send8( ) permet au NAO d'émettre un Octet.
La valeur de l'octet ici est passée en paramètre (data), sa valeur n'a pas de réelle importance, seul la détection d'un signal émis nous intéresse.

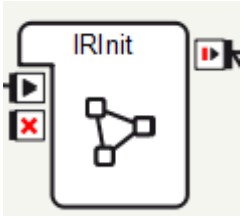
Au final ,lorsqu'un octet est émis par la fonction, la sortie de notre boite de commande est activée.

Nous disposons de deux fonctions permettant d'émettre et de recevoir un octet. La communication entre les NAOs est maintenant possible.

c) Problèmes rencontrés

Lors du test de nos deux fonctions, nous avons été confrontés à un problème sur notre réception. Le premier NAO envoyait bien son message mais le deuxième NAO ne recevait rien, bien que le détecteur infrarouge était sensible au signal (clignotement lorsqu'il détecte un message infrarouge)... En effet, il est possible de vérifier si un message a été envoyé ou reçu dans la mémoire du robot (ALMemory → InfraredIpAdressReceived)
Après plusieurs recherches dans la documentation technique et sur internet, nous en avons déduit qu'il y avait un problème d'initialisation des détecteurs, problème qui ne devrait logiquement pas avoir lieu d'être.
Nous avons donc réaliser le programme python suivant (tiré d'un exemple) permettant d'initialiser de manière « globale » le récepteur et émetteur infrarouge:

Initialisation de l'infrarouge:

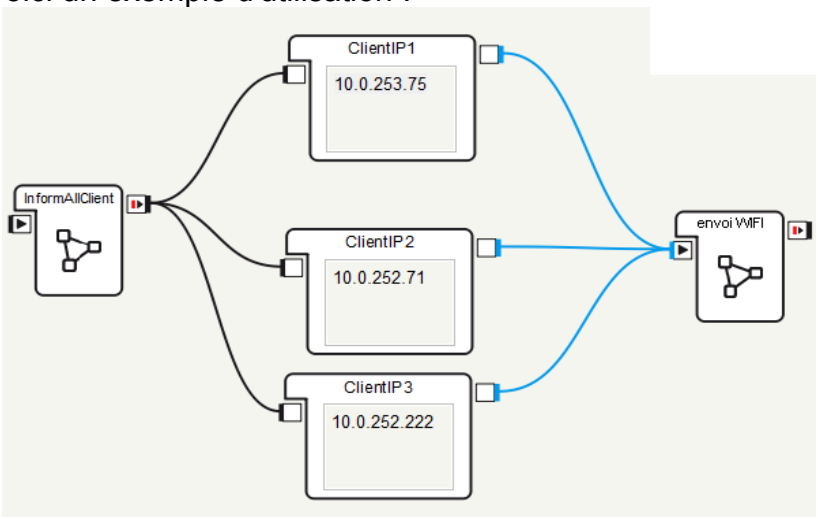


code python:

```
import naoqi
from naoqi import ALProxy

class MyClass(GeneratedClass):
    def __init__(self):
        GeneratedClass.__init__(self)

    def onLoad(self):
        #~ puts code for box initialization here
        pass

    def onUnload(self):
        #~ puts code for box cleanup here
        pass

    def onInput_onStart(self):
        #~ self.onStopped() #~ activate output of the box

        lirc=ALProxy("ALInfrared","127.0.0.1",9559)
        eventName = "InfraRedIpAdressReceived"
        pythonModule = ALModule("pythonModule")
        prox = ALProxy("ALMemory")
        prox.subscribeToEvent(eventName,"pythonModule", "pythondatachanged")
        lirc.initReception(10);
        self.onStopped();
        pass

    def onInput_onStop(self):
        self.onUnload() #~ it is recommanded to call onUnload of this box in a onStop
method, as the code written in onUnload is used to stop the box as well
        pass
```

Nous exécutons donc cette boite de commande avant celle permettant de recevoir un octet. Finalement la réception d'un octet émis fonctionne comme nous le désirions.
Notre méthode de communication n'est pas sans aucunes contraintes, en effet les robots sont obligés de se regarder (avec un certain degré de liberté) pour communiquer.
Notre cahier des charges ne nous imposant rien sur le proximité des robots et leur disposition.
La communication par WIFI permettrait de résoudre cette contrainte, nous avons demandé a la société Aldebaran un exemple d'utilisation de celle-ci.
Leur réponse a malheureusement été trop tardive pour adapter nos programmes avec cette méthode de communication cependant nous savons maintenant l'utiliser.

Voici un exemple d'utilisation :



Chaque adresse IP représente un NAO, ces trois adresses sont passées en paramètre pour la boite de commande « envoi WIFI ».

Voici le programme python associé à cette boite de commande :

```
class MyClass(GeneratedClass):
    def __init__(self):
        GeneratedClass.__init__(self)

    def onLoad(self):
        #~ puts code for box initialization here
        pass

    def onUnload(self):
        #~ puts code for box cleanup here
        pass

    def onInput_onStart(self, strClientAddress ):
        remoteMem = ALProxy( "ALMemory", strClientAddress, 9559 );
        remoteMem.raiseMicroEvent( "sync_start", True );
        self.onStopped() #~ activate output of the box
        pass

    def onInput_onStop(self):
        self.onUnload() #~ it is recommanded to call onUnload of this box in a onStop method, as
the code written in onUnload is used to stop the box as well
        pass
```

Explication du programme:

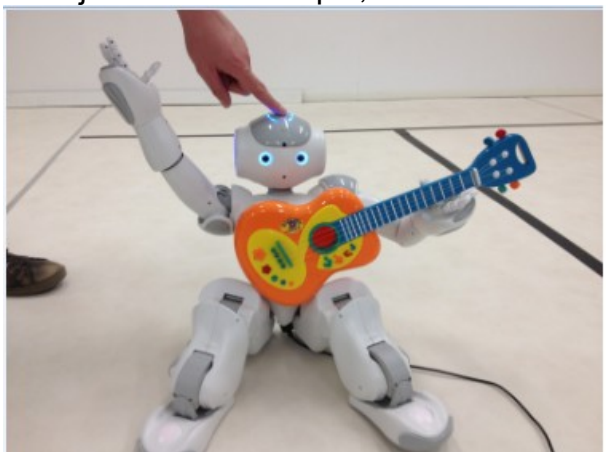
Les 2 NAOs étant connectés en WIFI sur un même routeur, il est possible d'échanger des informations.
Un NAO « serveur » envoie un message à un NAO « client », si un « serveur » reçoit un message il peut agir comme client.
Pour cela, nous utilisons les fonctions suivantes :
AlProxy (« ALmemory », adresse robot client , 9595) : permet de se connecter au NAO « client ».
raiseMicroEvent (Nom de l'événement , valeur que l'on envoie) : permet d'envoyer une valeur.

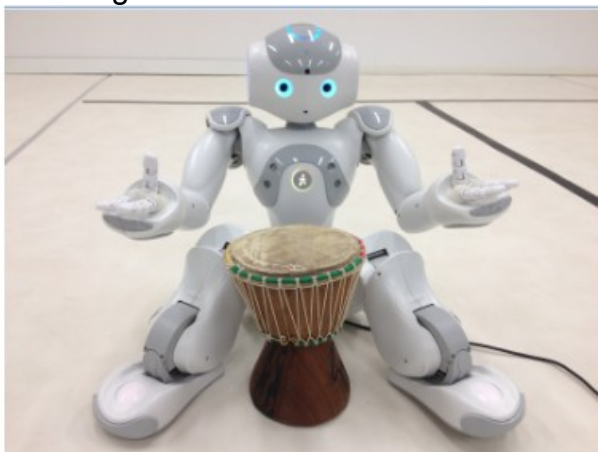
Finalment cette méthode permet un échange d'informations entre les NAOs sans aucune contrainte physique. Nous n'avons malheureusement pas eu le temps d'adapter tout notre programme à cette méthode de communication, il s'agit d'une amélioration à apporter pour de futur projets de coopération entre NAOs par exemple.

III – Nos programmes

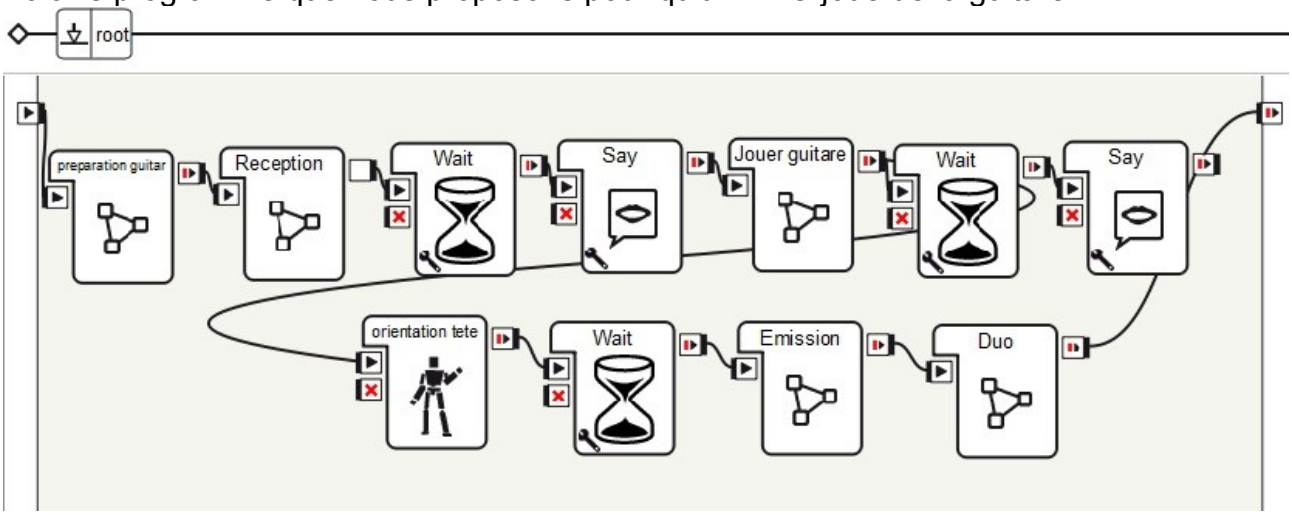
a) Le groupe de musique

Nous allons vous présenter les deux programmes que nous avons crées afin de faire jouer de la musique avec les NAOs.
Pour jouer de la musique, les NAOs utiliserons une mini guitare et un tambour :



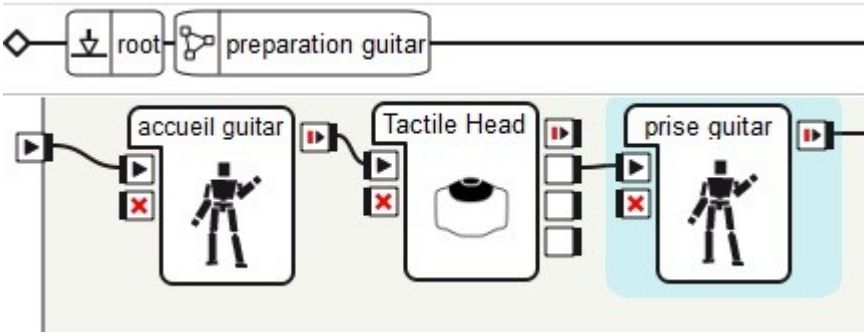


Un premier programme a été édité afin de permettre à un NAO de jouer de la guitare (respectivement un deuxième pour jouer du tambour), le tout en communication avec l'autre NAO.
Voici le programme que nous proposons pour qu'un NAO joue de la guitare:

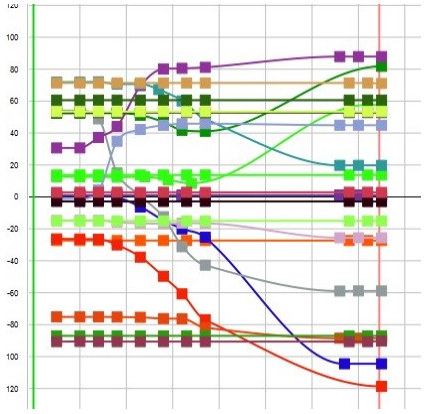


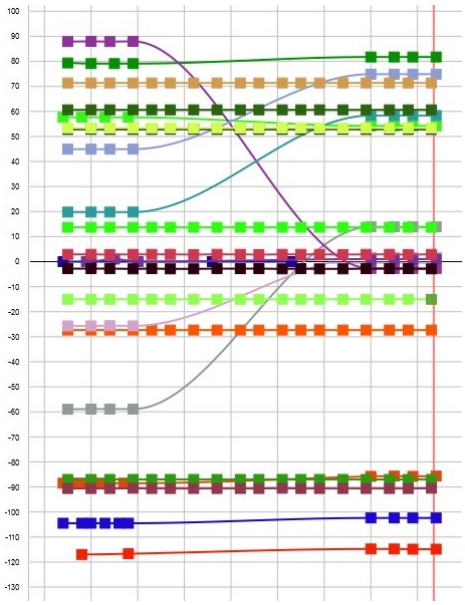
Comme vous pouvez le voir, le programme est composé de boites de commande, placées les unes à la suite des autres, ayant chacune un rôle bien spécifique. Nous allons vous en présenter de manière détaillée chaque étape.

La boite de commande préparation guitare est composée de sous-boites de commande:



« acceuil guitar » nous permet de placer le robot afin d'accueillir la guitare, les mouvements des moteurs sont visibles à droite.





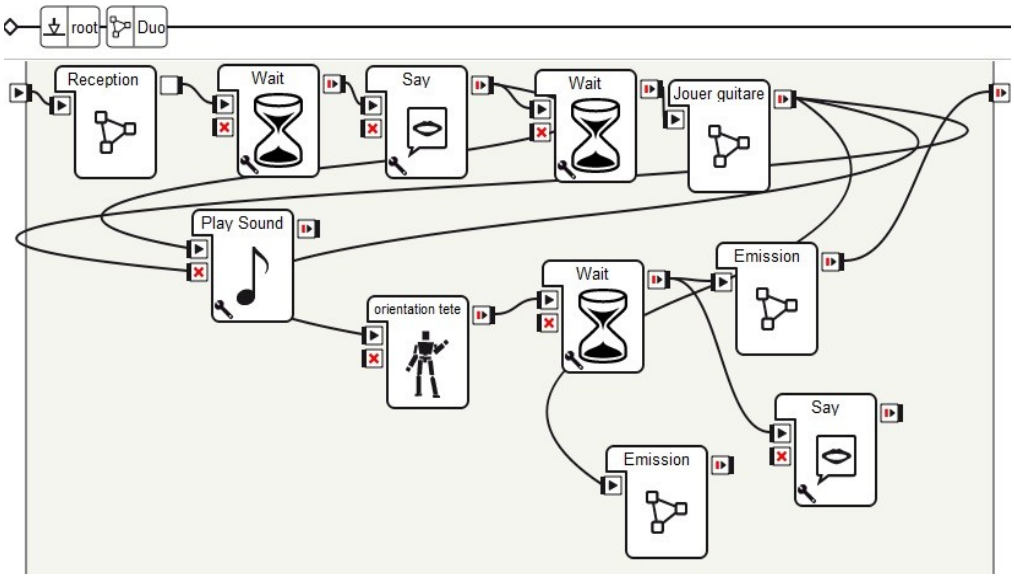
Une fois la guitare placée, il nous suffit d'appuyer sur le « Tactile Head » situé sur la tête du robot pour que celui-ci saisisse la guitare grâce au bloc « prise guitar » (mouvements des moteurs enregistrés à gauche).

Notre programme se compose ensuite d'une boite réception, nous utilisons dans cette boite de commande la réception infrarouge présenté précédemment. Le NAO est donc a cette étape en attente d'un signal infrarouge pour passer à la suite.

La prochaine boite de commande est une boite d'attente, nécessaire à la bonne harmonisation des mouvements. La boite de commande « Say » permet au NAO de « parler ».
La boite de commande « jouer guitare » contient les mouvements bien spécifiques permettant au NAO de gratter les cordes de la guitare, autrement dit d'émettre des notes de musique. Pour enregistrer ces mouvements nous avons enregistré différentes positions

intermédiaires afin d'optimiser le mouvement.
« orientation tête » permet au NAO de regarder son voisin pour un éventuelle communication (contrainte de notre méthode de communication)

La boite de commande « duo » permet de faire jouer les deux NAO simultanément:

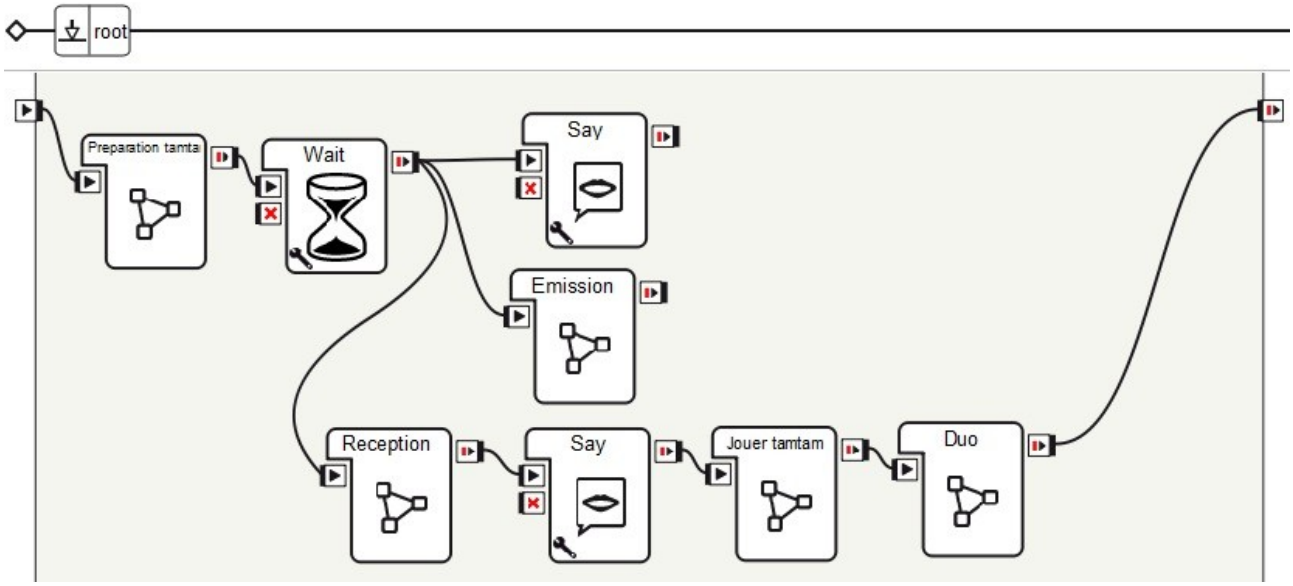


On retrouve dans cette boite de commande toutes les fonctions nécessaires à la communication afin de synchroniser les deux NAO, le bloc « jouer guitare » est le même que celui que nous avons vu précédemment. Le bloc « PlaySound » permet de joueur un morceau de musique se trouvant sur l'ordinateur. Les NAO vont donc jouer en playback pour le reste du programme.

Les instruments de musique n'étant pas spécialement conçus pour le NAO il a été très difficile de les adapter au NAO, en particulier la guitare
Les principaux problèmes rencontrés ont été matériels, nous avons cherché la meilleure position pour adapter la guitare au NAO. Suite a cette position, nous avons commencé la programmation, les mouvements du NAO sont cependant limités pour ne pas faire tomber la guitare.

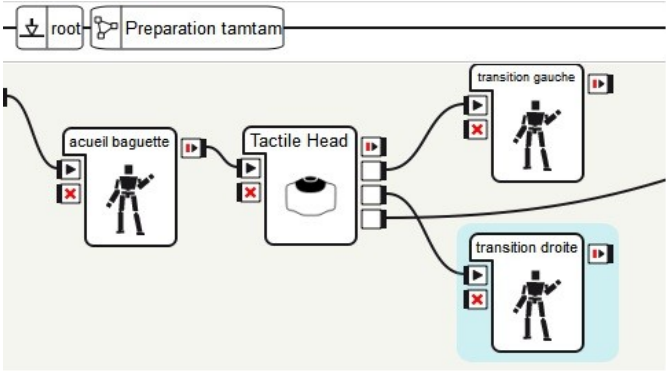
Nous avons également rencontré un problème avec les doigts du NAO, ceux-ci ne sont conçu que pour s'ouvrir et se fermer afin d'attraper un objet. Il nous à manqué quelques millimètre afin que le NAO touche les cordes (Problème résolu en ajoutant un embout au bout des doigts du NAO). Les notes de musique obtenues ne sont malheureusement pas aussi propres que si une personne reproduisait les mêmes mouvements, ceci est du à la souplesse des doigts du NAO associée à la rigidité des cordes...

Passons maintenant au second programme permettant au robot de jouer du tambour:



On retrouve les boîtes de commande d'émission et de réception Infrarouge présentées précédemment (clé de notre projet) . Les boites de commande « préparation_tambour », « jouer_tambour » et « duo » renferme comme nous l'avons expliqué précédemment des mouvement enregistrés, ici encore un long travail sur l'optimisation de chaque position des moteurs dans le temps a été fourni.

Le bloc de commande « preparation_tambour » permet a l'utilisateur de donner les baguettes au NAO.
Le robot tend les deux bras afin d'accueillir les baguettes, l'utilisateur les lui présente ensuite. nous utilisons les détecteurs frontaux afin d'indiquer au NAO qu'il peut saisir la baguette.



Comme vous pouvez le remarquer, l'évolution dans le temps de nos programmes (guitare et tambour) est géré par la réception d'un message infrarouge. La synchronisation de mouvement est ainsi possible. Au final, nos NAO utilisent les instruments pour jouer de la musique, chacun leur tour dans un premier temps et en duo ensuite.

b) La chorégraphie

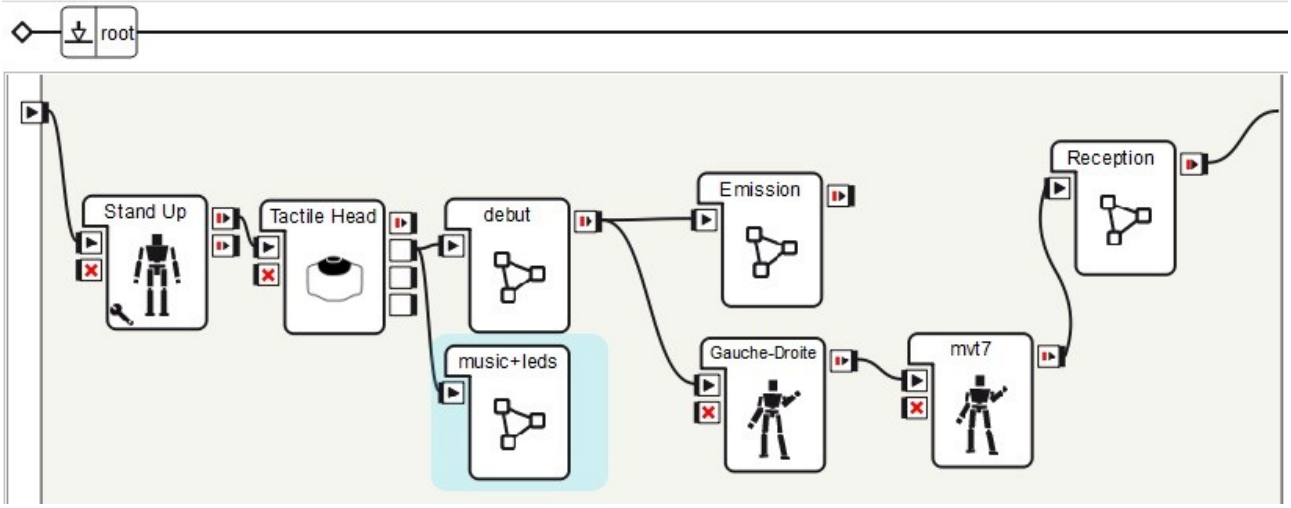
Une fois le groupe de musique opérationnel, nous avons voulu réaliser une chorégraphie avec les deux NAO, la synchronisation des NAOs sera une fois de plus gérée par l'émission et la réception infrarouge.
Nous avons choisi d'enregistrer différents mouvements plus ou moins périlleux , sollicitant un bon équilibre du NAO (voir annexe).
La réalisation des courbes traduisant la position dans le temps des moteurs nous a demandé beaucoup de travail. En effet, il a fallu à chaque fois trouver différents points intermédiaires pour permettre au NAO de tenir en équilibre, le plus souvent sur un seul pied.

Le déroulement de scénario choisi est le suivant:

Les deux NAOs sont en position de départ sur une même ligne face à l'utilisateur.
NAO1 commence ses enchaînements, NAO2 attend le signal de départ qu'il recevra via l'infrarouge du NAO1.
Après ses enchaînements, NAO1 émet un signal à NAO2 afin qu'ils réalisent tous deux un mouvement synchronisé, à la fin de ce mouvement, NAO2 fini ses enchaînements et NAO1 attend le signal de NAO2 pour couper la musique.

Nous avons donc pour cela créer deux programmes, un pour chaque NAO que nous allons vous présenter.

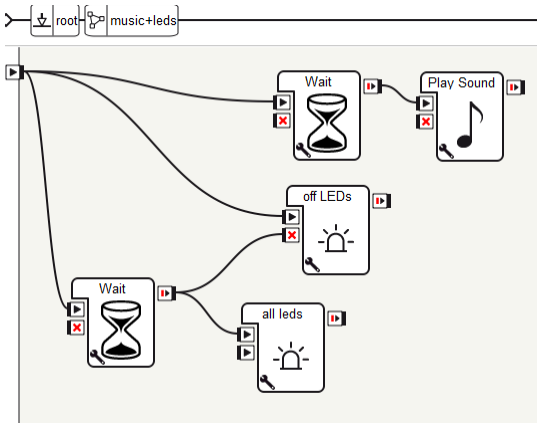
Programme choregraphie_NAO1:



Nous utilisons la boîte de commande « stand-up » et tactile head afin de mettre le robot en position debout et de l'avertir du début du programme (nécessaire pour placer les robots avant de commencer).

Les boites de commande « debut » et « music+leds » s'exécutent ensuite en parallèle.

« music+leds » permet le lancement de la musique choisi pour la chorégraphie (générique de du film TRON l'héritage) grâce à la boite « Play Music », l'extinction et l'allumage de toutes les leds à un moment donné.



Pour donner un coté plus vivant à notre chorégraphie et en accord avec la musique, nous avons décidé d'allumer toutes les Leds du robot d'une même couleur, nous avons pour cela créer deux codes python réalisant cette tache (deux boîtes de commande, « Off LEDs » et « Eyes_LEDS »). Nous utilisons les fonctions présentes dans la bibliothèque de fonctions NAO.

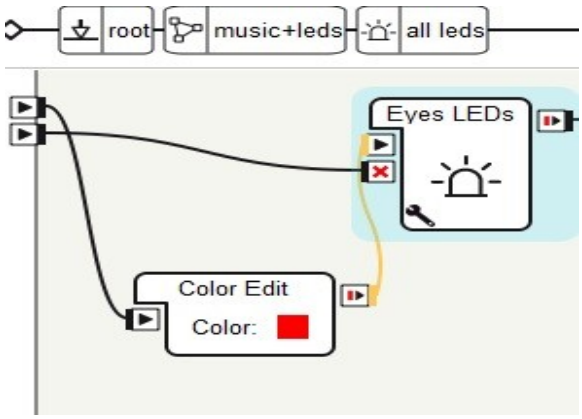
Code python de off LEDs :

Script Editor

test | off LEDs

```
class MyClass(GeneratedClass):
    def __init__(self):
        GeneratedClass.__init__(self)

    def onLoad(self):
        self.ids = []

    def onUnload(self):
        #~ puts code for box cleanup here
        pass

    def onInput_color(self):
        leds = ALProxy("ALLeds","127.0.0.1",9559)
        leds.off("AllLeds")


    def onInput_off(self):
        self.onDone() # activate output of the box
```

Find:



Code python de Eyes_LEDS :

Script Editor

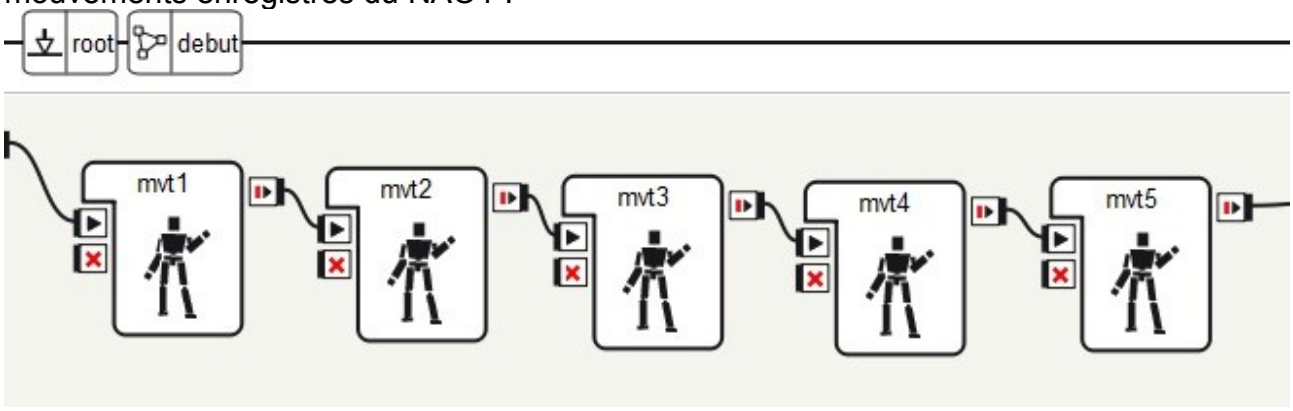
Eyes LEDs

```
class MyClass(GeneratedClass):
    def __init__(self):
        GeneratedClass.__init__(self)

    def onLoad(self):
        self.ids = []

    def onUnload(self):
        #~ puts code for box cleanup here
        pass

    def onInput_color(self, p):

        id = ALLeds.post.fadeRGB( "RightFootLeds",
256*256*p[0] + 256*p[1] + p[2],
self.getParameter("Duration (s)"))
        id = ALLeds.post.fadeRGB( "LeftFootLeds",
256*256*p[0] + 256*p[1] + p[2],
self.getParameter("Duration (s)"))
        id = ALLeds.post.fadeRGB("FaceLeds",
256*256*p[0] + 256*p[1] + p[2],
self.getParameter("Duration (s)"))
        id = ALLeds.post.fadeRGB( "ChestLeds",
256*256*p[0] + 256*p[1] + p[2],
self.getParameter("Duration (s)"))

        self.ids.append(id)
        ALLeds.wait(id, 0)
        self.ids.remove(id)
    def onInput_stop(self):
        a=0
        self.onDone() # activate output of the box
```

Ln 15 Find:

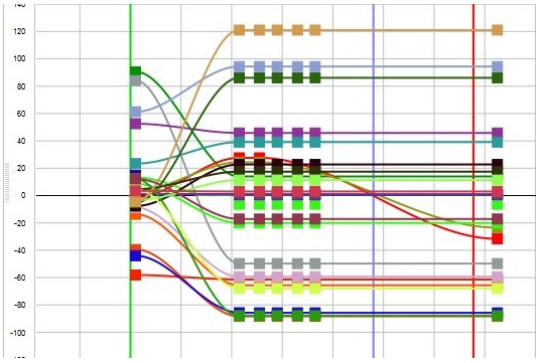
Explication des programmes :
-leds.off( ) : fonction permettant d'éteindre toutes les Leds du robot.
-ALLeds.post.fadeRGB ( ) : fonction permettant d'attribuer une couleur à une Led.
Nous passons en paramètre la couleur souhaitée pour toutes les Leds du robot (rouge sur l'exemple). Chaque couleur possède trois composantes, rouge, verte et bleue correspondant à p[0], p[1], et p[2]. Nous utilisons ensuite cette fonction pour toutes les Leds du robot (pied droit et gauche, les yeux et la led centrale).

Venons-en maintenant à la boite de commande « début », elle se compose des différents mouvements enregistrés du NAO1 :

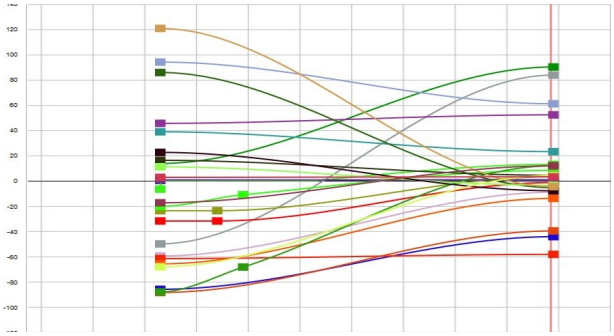


Nous avons donc réalisé 5 mouvements différents, comme dit précédemment nous avons choisi de donner au NAO des mouvements plus ou moins complexes, le but étant d'arriver à une position finale (en passant par plusieurs étapes intermédiaires) et de revenir à notre position initiale (debout) afin de commencer le prochain mouvement.
Pour vous donner une idée de ce que représente chaque mouvement, voici les différentes positions des moteurs dans le temps au cours de ceux-ci :

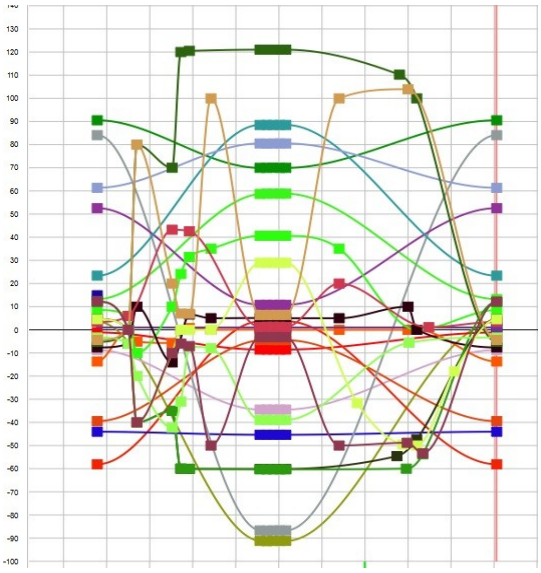
« mvt 1 »



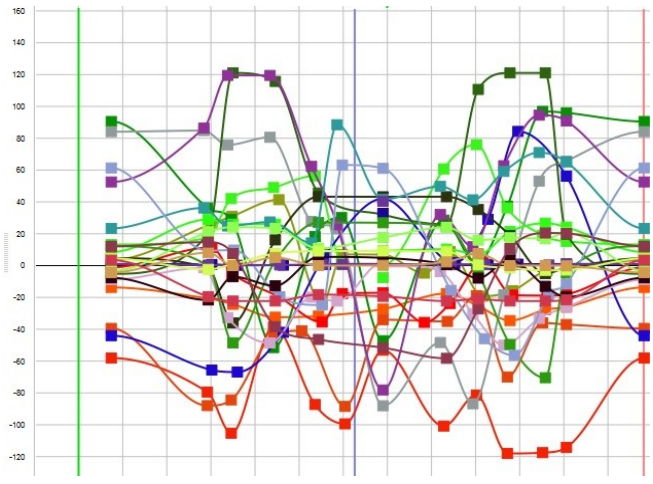
« mvt 2 »



« mvt 3 »

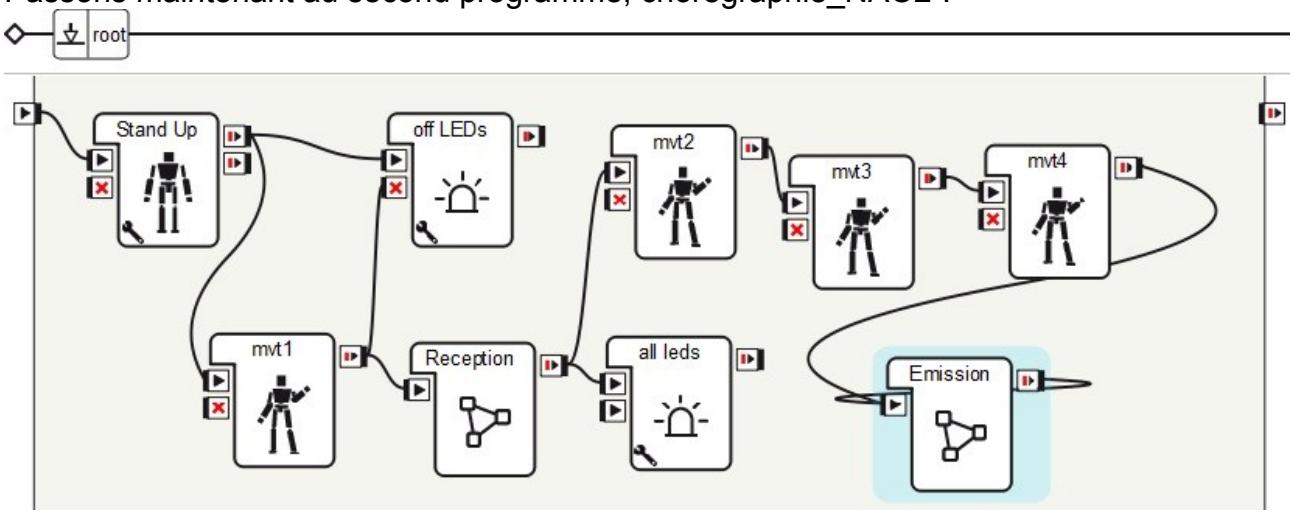


« mvt4 »



Les courbes obtenues sont représentatives de la complexité du mouvement effectué. Chaque point, pour chaque moteur a été ensuite retravaillé et modifié afin d'obtenir un mouvement le plus harmonieux possible et surtout faire en sorte que le NAO reste en équilibre afin d'éviter toute chute.
Un dernier point important du programme que nous proposons est le mouvement enregistré dans la boîte de commande « gauche-droite ». ce mouvement va être exécuté en même temps par les NAOs au cours de la chorégraphie (synchronisation par infrarouge). En effet une boîte de commande « Emission » va venir s'exécuter en parallèle.
Une fois ce mouvement terminé, on entre dans une boîte de commande de type « Reception », le NAO attend un signal pour terminer son programme (autrement dit couper la musique qui est toujours en cours d'exécution).

Passons maintenant au second programme, chorégraphie_NAO2 :



Les mouvements présents dans ce programme ont déjà été présentés précédemment, nous utilisons en plus les boîtes de commande « emission » et « reception » pour la communication ainsi que le programme permettant d'éteindre et d'allumer toutes les Leds du robot.

Déroulement du programme :
Le NAO va éteindre ses LEDs et se mettre en position pour attendre le signal. Un fois le signal reçu il va allumer toutes ses LEDs (en bleu) puis enchaîner ses mouvements. En fin de programme, Le NAO envoie un signal au premier NAO afin de l'avertir qu'il a terminé.

Nos deux programmes vont permettre aux NAOs d'effectuer une chorégraphie tout en communicant afin de se synchroniser sur l'exécution de mouvements.
Les problèmes rencontrés au cours de cette partie ont été principalement sur l'équilibre du NAO lors de la réalisation des mouvements, un long travail d'optimisation des courbes a eu raison de ses problèmes. Nous avons également essayer d'enchaîner nos mouvements le plus vite possible, cependant ceux-ci restent assez lents. Un compromis est nécessaire lors de l'exécution d'un mouvement entre l'équilibre du robot et la vitesse d'exécution du mouvement . De plus la complexité de nos mouvements est très éprouvante pour les différentes articulations du robot, celles-ci chauffent très vite.

Malheureusement le NAO n'est pas équipé d'un système de ventilation performant, ce problème associé à l'autonomie de la batterie nous a fait perdre un temps non négligeable, néanmoins le NAO reste un robot complet aux multiples possibilités de programmation même si des améliorations sont encore nécessaire.

c) Tutoriel d'utilisation de nos programmes

Nos programmes étant disponibles, nous proposons un petit tutoriel d'utilisation.

Groupe de musique : guitar_communication.crg
Tambour_communication.crg

Chorégraphie : choregraphie_NAO1.crg
choregraphie_NAO2.crg

Pour tous nos programmes la position initiale du NAO est assise.

Groupe de musique

NAO1 (guitar), NAO2 (tambour)
Mettre les 2 NAOs face à face en position assise.
Commencer par charger le programme dans NAO2

NAO1:

-Lancer Choregraphe - se connecter à NAO1 (Ip: 192.168.2.10x).
-Ouvrir le projet "guitar_comunication".

-Asservir le robot.

-Lancer le programme.
-Placer la guitare sur NAO1.
-Appuyer sur la capteur tactile de son front.

NAO2:

-Lancer Choregraphe.
-Se connecter a NAO2 (Ip: 192.168.2.10x).
-Ouvrir le projet "Tambour_comunication.

-Asservir le robot.

-Lancer le programme.
-Mettre la baguette dans sa main droite (appuyer sur le capteur tactile de devant).
-Mettre la baguette dans sa main gauche (appuyer sur le capteur tactile du milieu).
-Appuyer sur le capteur tactile de derrière pour lancer le programme.

La chorégraphie:

NAO2 :

-lancer chorégraphe.
- se connecter à NAO2 (Ip: 192.168.2.10x).
-ouvrir le projet "choregraphie_NAO2".

-asservir le robot.

-lancer le programme.

NAO1 :

-lancer chorégraphe.
- se connecter a NAO1 (Ip: 192.168.2.10x).
-ouvrir le projet "choregraphie_NAO1".

-asservir le robot.

-lancer le programme.
-appuyer sur le détecteur de devant pour démarrer la danse.

Conclusion

Le projet Coopération entre robots humanoïdes avait pour but de créer un groupe de musique et une chorégraphie avec 2 NAOs, le tout en échangeant des informations afin que les NAOs puissent être synchrones sur différents mouvements.

Les programmes que nous proposons respectent le cahier des charges.
Nous avons répondu à la problématique de la communication en utilisant l'infrarouge du robot, une amélioration par communication WIFI est également possible afin de permettre une communication sans aucune contrainte.

Pour la gestion du groupe de musique et de la chorégraphie, nous avons principalement utilisé les outils d'enregistrement de mouvement proposé par Chorégraphe avec un long travail d'optimisation des différents mouvements afin de contourner les différents problèmes d'échauffement d'articulations et d'équilibre par exemple.

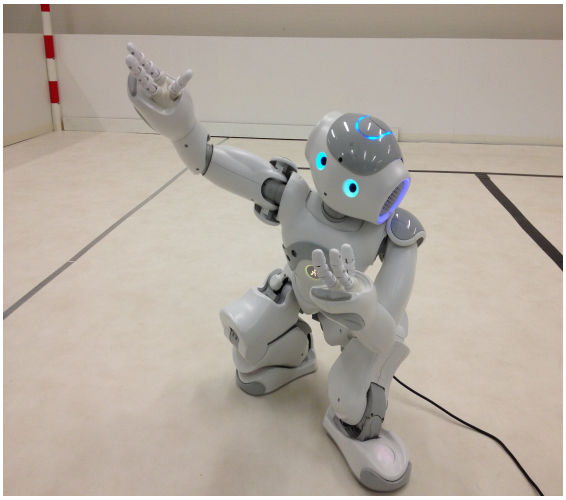
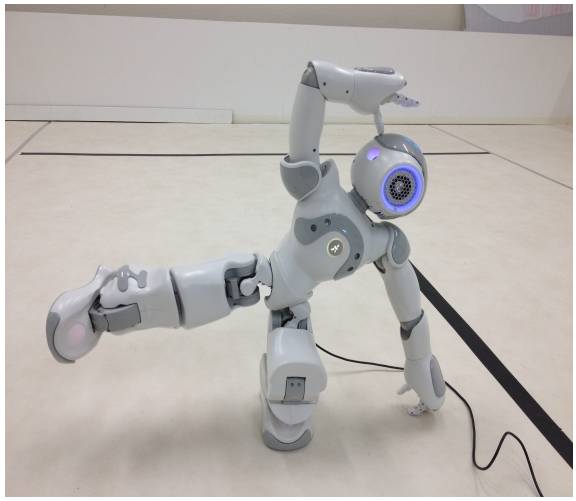
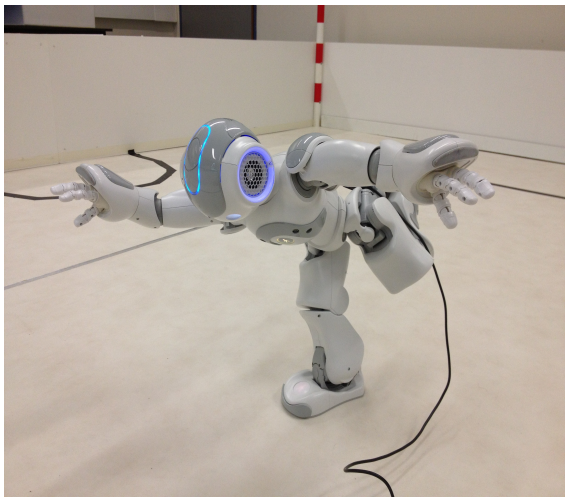
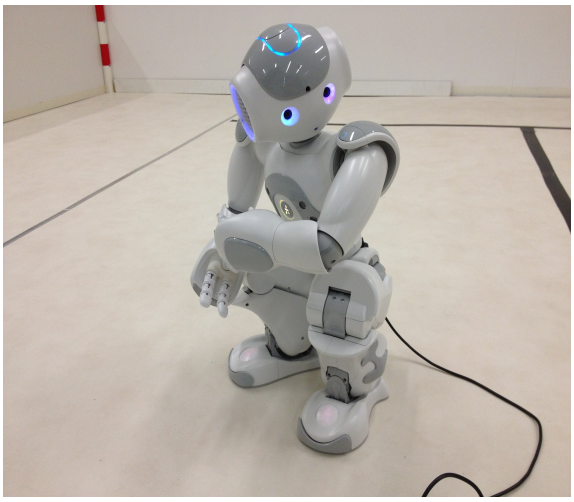
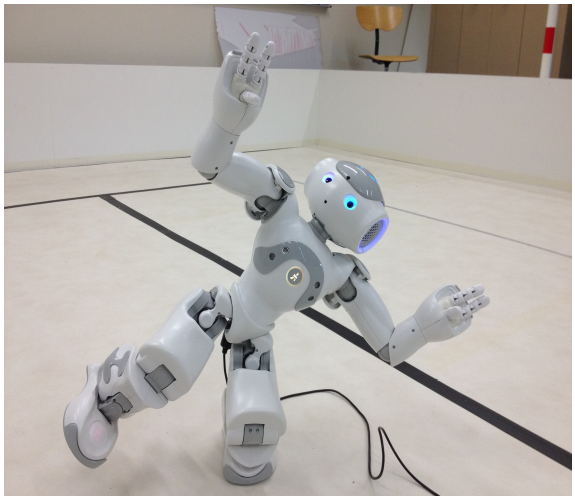
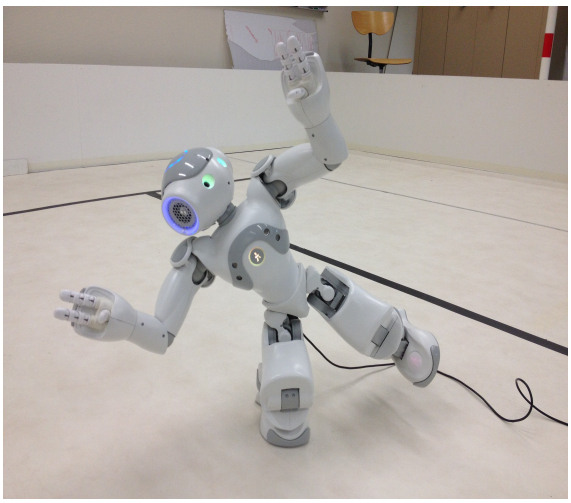
Ce projet nous a surtout permis d'étudier un système robotisé complexe tel que le NAO ainsi que l'étude des différents outils disponibles afin de répondre à un cahier des charges.

Le projet dans son ensemble s'est bien déroulé, la programmation d'un NAO restant très ludique, tout comme le cahier des charges. Cependant nous pensons que des améliorations sont encore à effectuer sur le NAO ainsi que sur le logiciel de programmation Chorégraphe, en effet nous avons rencontré de nombreux problèmes (autonomie, échauffements des articulations, problèmes de connections, de versions de logiciels, …)

En conclusion nous gardons une expérience positive de ce projet, très enrichissant. Il nous a sensibilisé sur les différents contraintes en programmation d'un robot mobile (équilibre, échauffement, autonomie). Nous avons essayé de respecter au mieux le cahier des charges demandé en utilisant les différents outils fourni par le constructeur et en essayant d'apporter une solution à chaque problématique...

Annexe

Voici les différents mouvements réalisé lors de notre chorégraphie :

Remarque : Pour arrivé à un de ses six mouvements, il est nécessaire de passer par différentes étapes intermédiaires afin de garder un équilibre correcte du NAO.