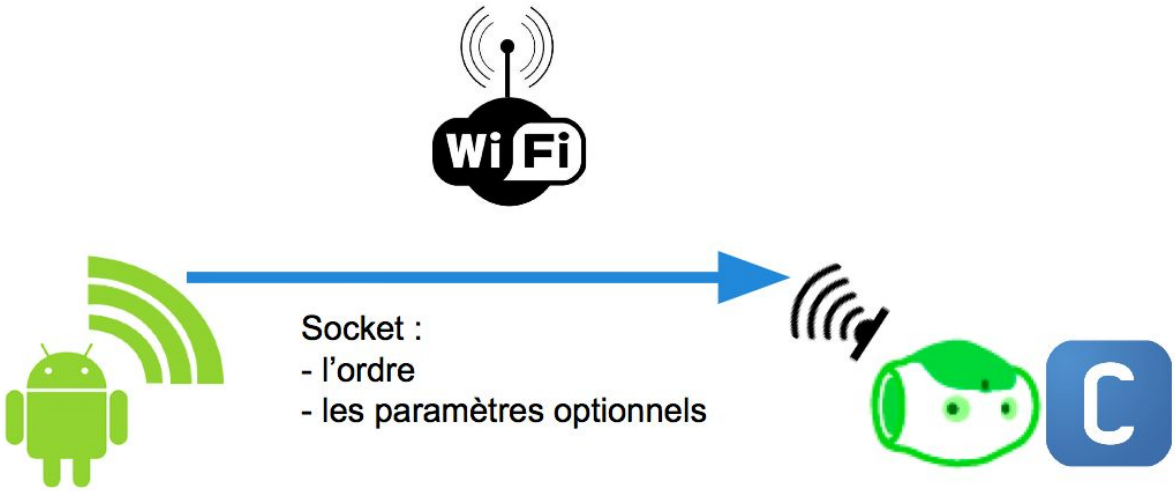
NAO Remote Controller

Le projet NAO était un sujet libre. Après quelques temps de réflexion il nous a paru intéressant d'essayer de prendre le contrôle de NAO à distance à la manière d'un drone. L'implémentation d'un serveur gérant un protocole de communication via Chorégraphe n'ayant pas présenté de difficultés majeures, nous souhaitions profiter du projet pour se familiariser avec de nouvelles technologies.

Nous avons donc d'abord développé un client lourd avec JavaFX; le remplaçant de Swing qui était jusqu'à présent la bibliothèque officielle d'Oracle pour le développement d'interfaces graphiques en Java.

Les applications smartphones étant au goût du jour, nous avons aussi souhaité créer une application Android permettant de télécommander le robot NAO.

Les deux systèmes communiquent grâce à une connexion WiFi. Vous trouverez ci-dessous un schéma de notre infrastructure.



Vous trouverez ci-dessous des précisions sur le serveur NAO créé grâce à l'IDE Choregraphe. Ensuite nous apporterons des explications sur les clients Android et Java. Enfin, nous aborderons les axes d'amélioration possibles

I. Le serveur Choregraphe

La création du serveur sous Choregraphe nous a permis de découvrir les problématiques liées à la robotique. En effet, avant de créer la version finale, nous avons commencé par demander à NAO d'épeler un mot. Ce dernier prononçait bien toutes les lettres du mot mais dans un ordre incorrect. La solution fut de temporiser la sortie des lettres avec un timer de 5 secondes. Cette expérience nous a beaucoup sensibilisé sur la gestion de la chronologie des actions.

Par la suite, nous avons étudié comment déclencher l'écoute d'une socket. Les ordres communiqués à NAO s'opèrent par le biais de cette dernière. Le modèle des sockets n'a pas été repris, nous avons opté pour une forme concise :

UID#Order#paremeter1#paremeter2#paremeter3...

UID : l'identifiant de l'émetteur. Ceci pourrait être repris pour limiter le nombre de connexions grâce à l'UID. Nous avons conscience que ceci peut être un axe d'amélioration explorable.

Order : l'action que NAO effectuera. Les actions sont indiquées en anglais. On retrouve par exemple "sayText", "sitDown", "standUp" et "walk".

Parameters : optionnel selon l'Order. Par exemple, pour la commande "walk", le paramètre 1 correspond à l'axe des X (aller vers l'avant ou l'arrière), le paramètre 2 à l'axe des Y (se décaler à droit ou à gauche) et le paramètre 3 pour l'axe de rotation (en positif pour tourner à gauche et en négatif pour tourner à droite).

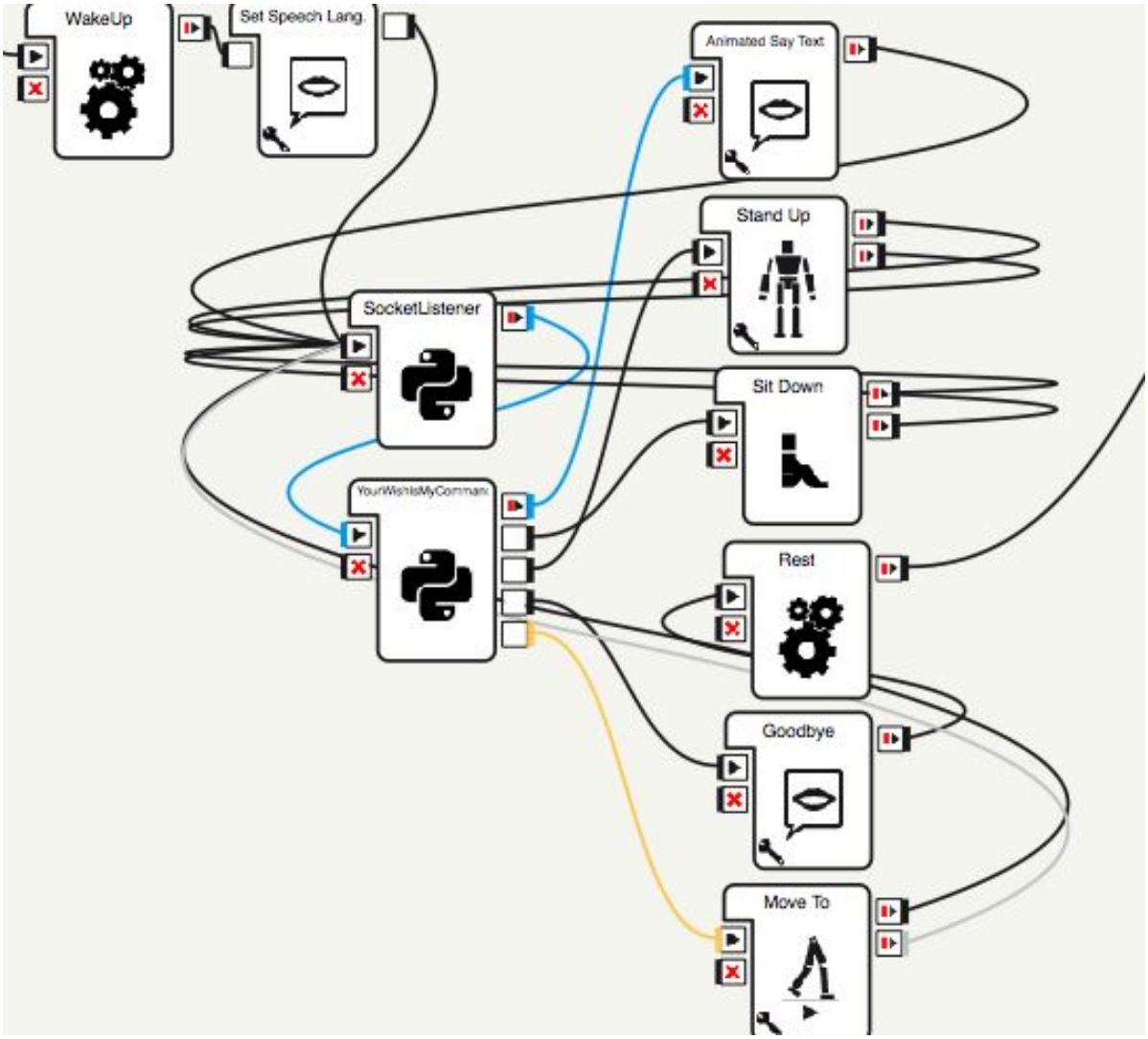
```
def onInput_onStart(self):
    # Set l adresse IP et le port
    self.HOST="" # prend l adresse IP de la machine par défaut
    self.PORT=1234
    # New socket
    self.socketOrder = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    self.socketOrder.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    self.socketOrder.bind((self.HOST, self.PORT))
    self.socketOrder.listen(1)
    self.connectionOrder, self.HOST = self.socketOrder.accept()
    self.logger.info("Connexion possible :"+str(self.connectionOrder)+":"+str(self.PORT))

    # Get socket message : UID, order, params
    total_data=""
    while 1:
        data = self.connectionOrder.recv(1)
        if not data:
            break
        else:
            if data=='$':
                p=total_data.strip(" \t\n\r").split("#")
            else:
                if len(total_data)<255:  # limite du data buffer
                    total_data+=data

    self.onStopped(p) #activate the output of the box
    pass
```

Nous avons rencontré des problèmes liés à la gestion de l'ouverture et la fermeture des sockets. A la fin de l'exécution de l'ordre, nous ne pouvions pas envoyer une nouvelle commande et Choregraphe ne fonctionnait plus. Nous faisons alors appel à la fonction onUnload dans laquelle nous fermons la socket créée. Cette fonction étant appelée systématiquement à la fin du traitement de l'ordre, nous n'avons plus de problème.

```
def onUnload(self):
    # Destructeur pour recommencer l'écoute
    self.socketOrder.close()
    self.connectionOrder.shutdown(socket.SHUT_WR)
    self.connectionOrder.close()
    pass
```

Nous avons aussi eu besoin d'un temps d'adaptation afin de maîtriser la technique permettant de passer des paramètres entre les box de Choregraphe. De plus, les types de variables sont légèrement différents. Nous avons donc étudié les spécificités de chacun dans le but de choisir le type le plus judicieux possible.

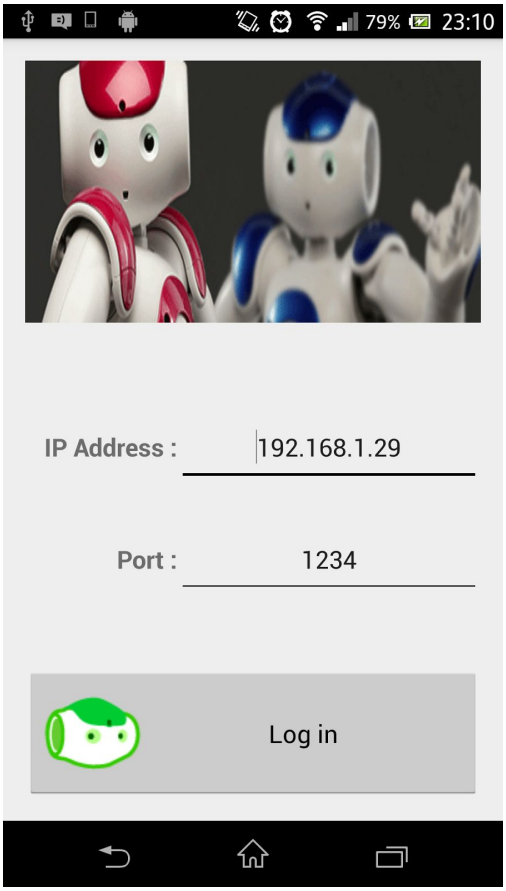
Le résultat final est le suivant :



II. La télécommande Android

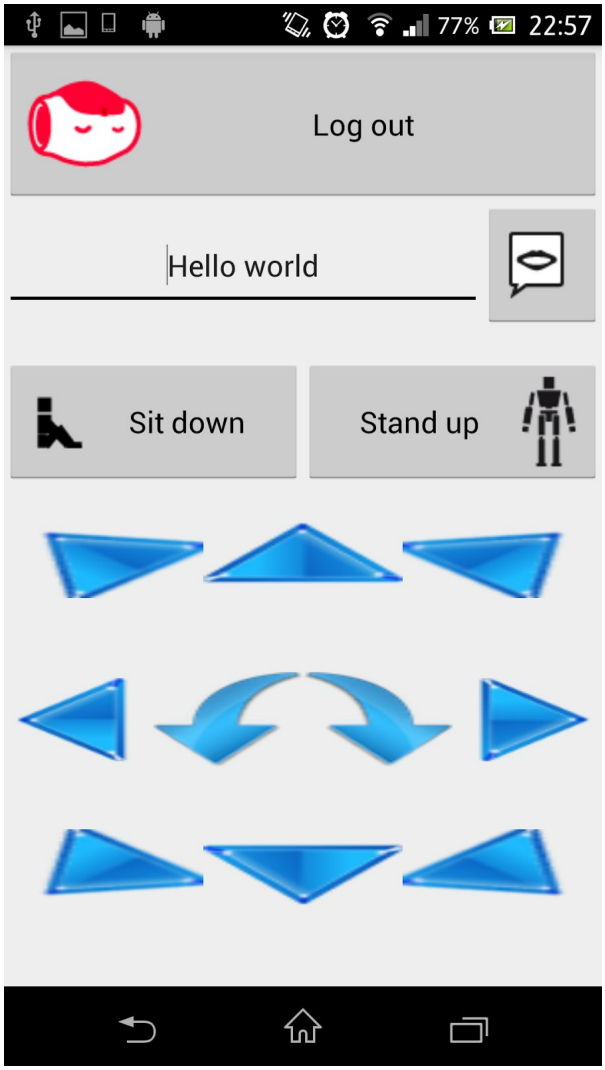
A. Télécommande Android

L'application Android est en fait une télécommande. La première étape est de repérer à quelle adresse IP NAO est accessible.



Une fois la connexion établie, une interface permet de choisir quelle action NAO doit exécuter. On peut le demander de dire un texte précis, de s'asseoir ou de se lever et de se déplacer. Un bouton de déconnexion est bien évidement disponible.

L’interface utilisateur est la suivante :

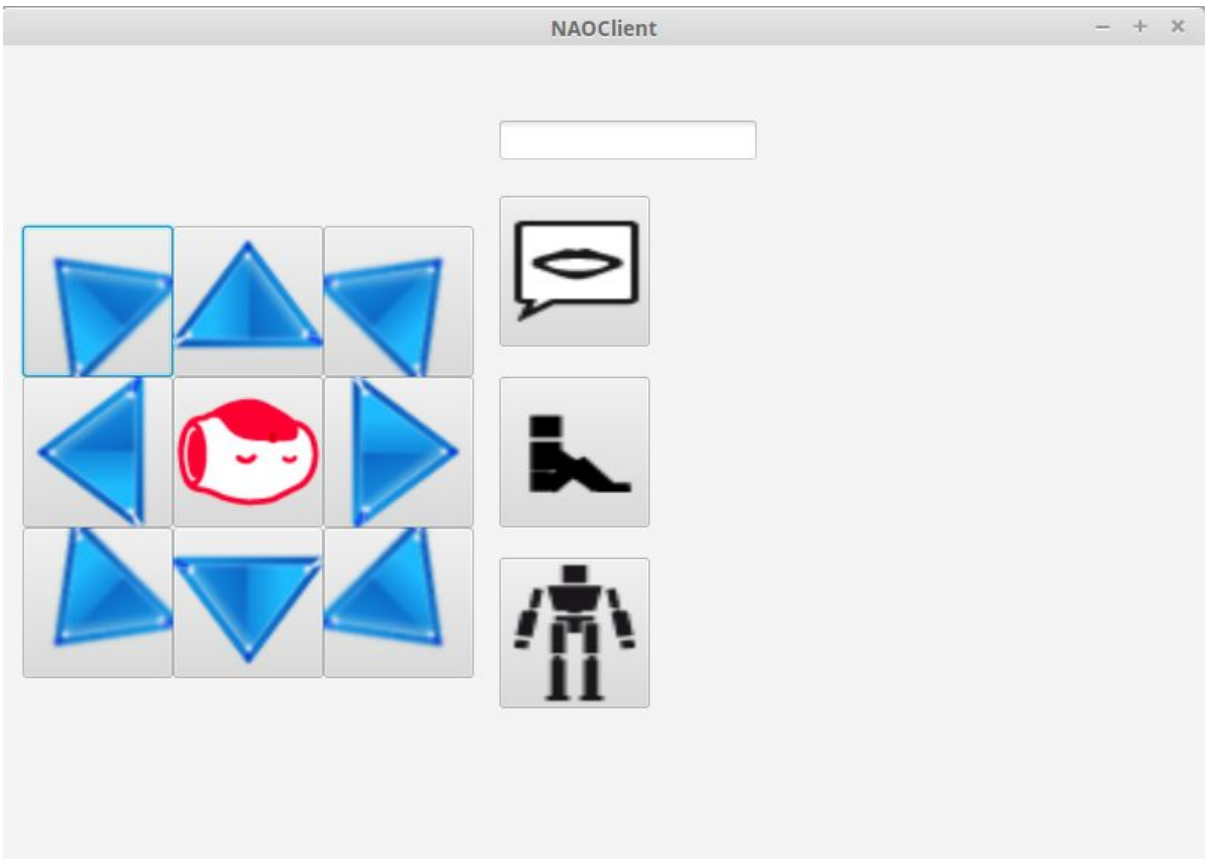


Vous avez à disposition le lien suivant pour visualiser une démonstration : http://kimsavinfo.fr/shared/NAO_remote_demo_1080p.mp4

B. Le client lourd

Les fonctionnalités sont les même, un premier écran de login permet de saisir les informations nécessaires à la connexion:

Une fois connecté, l'écran de contrôle s'affiche:



L'interface est moins user-friendly, mais le JavaFX présente des grosses difficultés quant à la compréhension de la gestion des enchaînement et l'architecture de l'application.

III. Les axes d'amélioration

Un axe d'amélioration possible concerne la connexion au robot NAO. L'émetteur d'ordre n'est pas analysé. Grâce à l'UID transmis, nous pourrions limiter les connexions à un utilisateur à la fois. De même, nous avons la possibilité de créer une whitelist. De cette dernière, seules les télécommandes habilitées seront autorisées à contrôler NAO. Cette fonction serait intéressante si NAO devient un robot domestique.

Si NAO se démocratise et si les foyers se fournissent de ce robot, la récupération du flux vidéo permettrait de transformer NAO en un robot de surveillance. Avec le contrôle à distance, on pourrait en plus lui demander de se déplacer si un objet est suspect.

Le projet NAO fut intéressant pour l'aspect robotique. Nous n'avions pas encore eu la chance d'être initiés à ce monde. Le fait de le découvrir et de le coupler avec une technologie d'actualité fut une riche expérience.

Johan HENIN
Amandine PALLAS
Kim SAVAROCHE