



GNUbiquity

Rapport n 3 – Semestre 8

Attié William
Bouly Enzo
Fath Thomas
Le Borgne Pierre
Rama Suntharasarma



*APP Robotique de Service 2017
Semestre 8*

Table des matières

Introduction :	3
Séance d'APP type	5
Compétences humaines développées	6
Détail technique :	7
Côté robot :	7
Partie Serveur	9
Le serveur et le robot de test	9
De l'ordre dans les fonctionnalités	10
Et ensuite ?	10
Transmission d'images	11
Développement de l'application Web :	13
Outils et méthodes de développement	17
Conclusion	17

Introduction :

Au début du semestre, le projet avait une structure et une version aboutie de l'interface graphique. Grâce à cet élément, nous avons pu partir sur une solide fondation. L'interface graphique a l'avantage de réunir dans un élément simple l'intégralité des fonctionnalités que devrait implémenter notre application finale. Ainsi nous avons pu progresser étape après étape pour essayer d'atteindre notre objectif tout au long de ce semestre.

Ainsi, nous avons pu effectuer les tâches suivantes :

- Finir de prendre en main le fonctionnement de Nao
- Implémenter les mouvements et la transmission vidéo
- Mettre en place une boîte à outil d'installation et de déploiement de l'environnement de programmation de Nao sur toutes les machines.
- Coordonner serveur, robot et tablette pour que tout fonctionne

Sur le plan humain et organisationnel, nous avons pu mettre en place la méthode scrum de développement de produit et avancer séance après séance avec une progression réaliste et maîtrisée. Ci-dessous quelques chiffres sur notre projet ce semestre:

- 14 séances de 4h à 5 élèves ingénieurs : 280h de travail pour l'aboutissement du projet
- Quelques chiffres depuis la création Github de notre projet :
 - o Plus de 8500 lignes de codes ont été ajoutées
 - o 90 commit
 - o Tous sont concentrés sur ce dernier semestre, comme en témoigne le graphique ci-dessous :



Le travail a toujours été séparé en trois groupes :

- Support de l'application client : Enzo et Pierre
- Déploiement et développement du serveur : William
- Prise en main et mise en place du robot : Thomas et Rama

Néanmoins, vers le milieu du projet, l'application se voyait déjà en phase d'amélioration, les fonctionnalités étant toutes opérationnelles et implémentées. Ainsi, Enzo a pu laisser libre cours à ses talents d'artistes en ajoutant une réelle identité graphique à notre application, pour que l'utilisateur puisse réellement se plonger dans un environnement confortable et intuitif et profiter de son expérience.

Pendant ce temps, Pierre a pu soutenir l'équipe travaillant sur le robot afin d'accélérer la prise en main des proxys et nous aider à faire plus rapidement le meilleur choix. Quant à William, ses connaissances poussées en informatique nous ont été d'une grande aide, d'autant plus qu'il a fait acte de présence auprès des deux autres équipes, pour nous aider à débloquer plus rapidement, tout en restant pédagogue.

Séance d'APP type

Les séances du matin ont toujours commencés à 7h30 à la cafétéria de l'école. Elles nous permettent de nous retrouver un moment ensemble avant d'entamer le travail. Nous y définissons les objectifs de la séance en fonction du travail restant à faire. De plus, ces moments en dehors de la salle d'APP nous ont permis de prendre le temps de mieux se connaître pour pouvoir travailler dans une ambiance agréable et conviviale.

Ces goûters ont été essentiels à notre projet, car nous sommes tous d'accord pour dire que les séances d'app où nous avons le plus avancés, sont également celles avec les meilleurs goûters !

Lors des séances tenues en après-midi, nous arrivions toujours avec des objectifs déjà fixés, notamment grâce à notre méthode scrum de développement de projet. Néanmoins, vers la mi-séance, réunissions pour parler des avancements : c'est alors le moment de débloquer celui qui a un souci en lui soumettant une nouvelle piste de recherche. Sinon, c'est également le moment parfait pour refixer nos objectifs à court et long terme.

Voici une brève introduction faisant part de l'organisation de notre équipe. Maintenant, nous allons vous présenter le travail technique qui a été effectué, les difficultés auxquels nous avons fait face pour arriver au bout de notre projet.

Mais avant d'entrer dans le détail, voici un bref aperçu des compétences humaines qui ont été développées tout au long des séances d'APP.

Compétences humaines développées

Au fil des séances et de l'avancement, chaque membre de l'équipe a pu acquérir des compétences humaines et organisationnelles qui les aideront dans chacun des projets auxquels ils prendront part. En effet, nous avons eu de longues discussions sur la gestion du projet et des différentes visions de chaque membre sur les solutions techniques possibles.

C'est ici la première compétence qui nous a été acquise, celle de se concerter sur un projet et prendre des décisions communes.

La deuxième compétence a été bien plus importante et prend une place plus importante dans l'aboutissement du projet. Elle concerne directement les compétences techniques de chacun des membres. Tout d'abord, nous nous étions répartis les tâches en fonction des affinités personnelles de chaque membre.

Ensuite, lorsque quelqu'un a un souci dans son travail, il devient important de tenir compte du facteur temporel et faire part de ses difficultés dès qu'il passe trop de temps sur son défi technique. C'est comme ça que le semestre dernier, nous avons perdu la moitié du semestre à se borner dans le choix du langage de programmation. Au final, nous nous sommes rangés sur le Python, un choix bien plus raisonnable et confortable que le C++.

Fort de cette leçon, nous avons beaucoup moins hésité ce semestre à partager nos savoirs et avoir un gain significatif de productivité.

Pour finir, nous n'avons heureusement pas eu affaire à des membres moins intéressés ou encore un élément fortement démotivé, nous imposant de recourir à la contrainte pour aller de l'avant. Les mises au points réguliers de l'avancement de chaque membre a été suffisant pour tenir tout le monde motivé et fidèle à la tâche.

Entrons maintenant dans le détail technique et les aboutissements de notre projet.

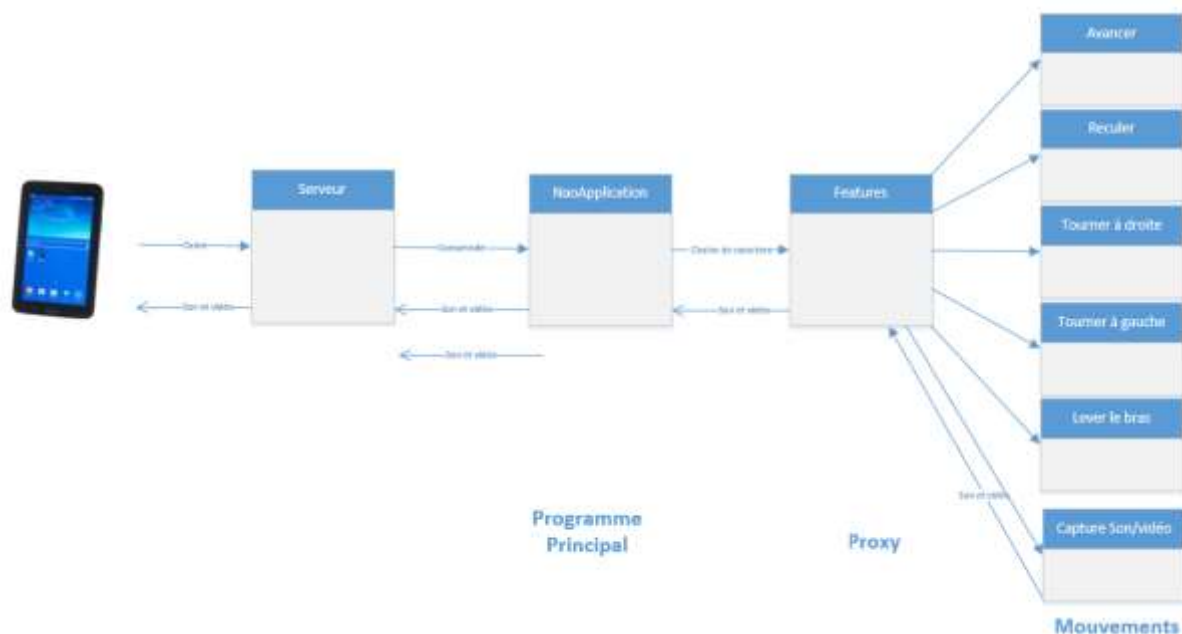
Détail technique :

Côté robot :

Dans notre souhait d'éviter toutes "NaoDependance", c'est-à-dire être indépendant du matériel, nous avons créé une interface afin de lier notre robot à notre serveur. C'est cette dernière qui faudra réécrire si nous changeons de robot, tout le reste restera identique.

Cependant cette interface est dépendante des fonctionnalités du robot. Nous avons une liste d'actions disponible au niveau du serveur, qu'on devra choisir d'implémenter ou non dans notre interface en fonction des capacités du matériel. Par exemple, ici nous avons utilisé un robot de type humanoïde, il peu par conséquence avancer et s'asseoir, si nous passons sur un robot de type galette nous pourrons implémenter la fonctionnalité avancer mais pas celle de s'asseoir.

Afin de répondre à notre cahier des charges établi au semestre 7 nous avons mis en place de nombreuses fonctionnalités sous formes de proxy.



Tous d'abord nous avons implémenté les déplacements et translation en X et Y ainsi que les rotations afin d'assurer les déplacements du robot. Ensuite afin de donner la possibilité au robot d'interagir avec le monde qui l'entoure nous avons ajouté une fonction permettant d'attirer l'attention et de poser une question, nous l'avons traduit sur notre matériel par un levé du bras.

Dans le but d'économiser la batterie et d'assurer une plus grande autonomie du robot nous avons implémenté une position de repos, pour notre cas c'est la position assis.

Les dernières fonctionnalités et les plus importantes ont été le retour vidéo et son. Pour le retour vidéo nous prenons plusieurs photos par seconde, en fonction des ressources processeur disponibles. Ensuite nous les envoyons au client, ce qui permet d'avoir une vision temps réel du robot. De plus nous pouvons orienter la tête du robot et donc la vision en fonction de l'endroit où l'utilisateur appui sur l'image de l'application web, pour cela nous récupérons la position absolu de la tête du robot puis nous calculons et ajoutons l'angle adéquate pour recentrer l'image sur le point d'appui de l'utilisateur.

Cependant nous rencontrons plusieurs problèmes matériel, le robot que nous utilisons (NAO) est programmé de manière que certains proxy sont prioritaire sur d'autres et ne peuvent fonctionner en même temps. Ce qui a pour conséquence de couper la vidéo lors de certaines autres actions comme s'asseoir et tourner. De plus le proxy audio ne peut pas fonctionner en même temps que celui de la vidéo. Il a donc fallu faire un choix entre conserver la vidéo ou avoir le retour du son.

Lors de nos tests nous avons remarqué que le son est extrêmement bruité par les bruits des ventilateurs du robot, il faudrait donc filtrer ce son. Niveau pratique il est plus intéressant de conserver la vidéo, notamment pour les déplacements. Nous avons fait de nombreuses recherches pour essayer de résoudre ces problèmes en vain, nous avons même vu qu'une Start up Suisse faisait un projet semblable au notre et pour résoudre ce problème ils ont installés un smartphone sur le robot avec un logiciel pour partager la vidéo et le son. Ceci qui prouve bien que les conflits de proxy sont difficile/impossible à résoudre.

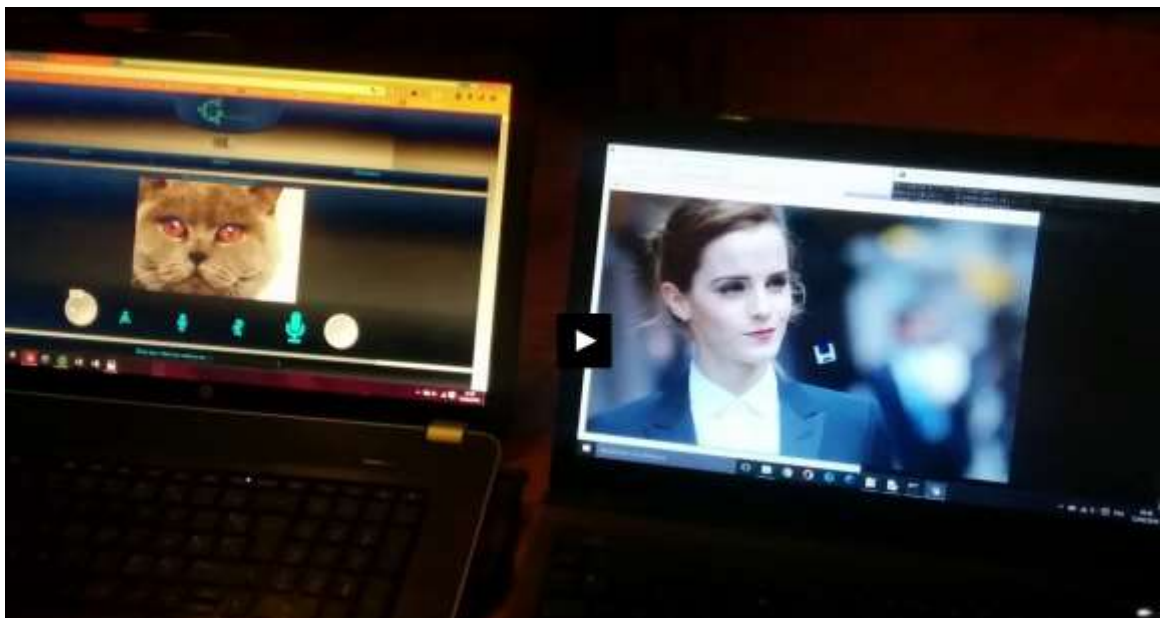
La conclusion que nous pouvons tirer est que la synchronisation par interface est une bonne solution pour éviter toutes dépendances du matériel et cela fonctionne correctement. Tous les problèmes rencontrés ici sont dû au matériel, en le changeant par un robot plus performant nous pourrions les résoudre. Pour effectuer tous changement de robot il suffit de reprogrammer l'interface que nous avons estimée à environ 40 heures de travail en fonction du robot.

Partie Serveur

Au début de ce semestre, certaines fonctionnalités étaient déjà implémentées. On envoyait l'interface web aux clients et on recevait les ordres donnés via l'interface.

Le serveur et le robot de test

Dans la mesure où notre projet n'est pas nao dépendant, on a créé un faux robot utilisant l'interface définie dans nos spécifications. Grâce à cela, les fonctionnalités client/serveur pouvaient avancer à leur rythme et être testées même si les fonctionnalités n'étaient pas encore présentes côté serveur/Nao. Le robot de test consistait en une fenêtre graphique qui affichait les ordres de posture, qui déplaçait un sprite à l'écran en fonction des consignes de mouvement et qui envoyait un flux vidéo de 15 images secondes (photos de chats).



Comme certaines fonctions de Nao étaient bloquantes, nous avons lancé le code qui contrôle nao dans un thread depuis le fichier interface. Nous avons ensuite codé un driver pour Nao en prenant pour base le faux robot et en remplaçant petit à petit ses fonctionnalités par celles qui étaient implémentées sur Nao. Ainsi nous avons à chaque instant un moyen de vérifier chacune des fonctionnalités.

De l'ordre dans les fonctionnalités

Nous avons souhaité intégrer en premier les fonctionnalités les plus faciles et les plus utiles afin d'avoir un livrable efficace le plus tôt possible.

Nous avons donc commencé par implémenter le déplacement des robots. En tant que serveur, je reçois les valeurs des joysticks dès que ceux-ci bougent. Ces valeurs sont normalisées par le client entre 100% et -100% sur chaque axe du joystick. Je transmets toutes ces valeurs à la méthode `motion` du robot.

Nous avons ensuite implémenté les positions du robot, qui, bien que moins utiles que le retour vidéo, étaient beaucoup plus faciles à réaliser. Nous avons donc réalisé ces fonctions. Le client envoie au serveur sur front descendant l'ordre de se mettre dans une position. Le serveur appelle alors sur le robot la méthode correspondant à la position (`setPositionIdle` pour l'état actif du robot, `setPositionCue` pour le fait d'attirer l'attention et `setPositionRest` pour l'état de repos du robot).

Une fois ces fonctionnalités réalisées, je me suis penché sur le traitement de la voix Utilisateur-> Nao, mais nous avons eu du mal à savoir quels types de données étaient utilisés par les différentes bibliothèques. Comme cette fonctionnalité menaçait de prendre beaucoup de temps pour des résultats mitigés, nous avons décidé d'implémenter la communication par `textToSpeech` : le client envoie une chaîne de caractères qui est transmise à la méthode `sayText(txt)` du robot.

Comme il ne restait plus rien de "simple" et utile à ajouter au projet, on a commencé à travailler sur la partie retour vidéo. Du point de vue du serveur, il demande juste une image au robot en continu. Le challenge se trouvait plutôt au niveau du robot, Nao dans notre cas. Il fallait lui faire prendre une image, la transmettre à l'ordinateur de contrôle puis la mettre à disposition du serveur dans un format adéquat. Pour ce faire nous n'avons eu d'autre choix que d'enregistrer l'image temporairement sur le disque de l'ordinateur de contrôle et de la rouvrir dans le bon format ensuite.

Et ensuite ?

Comme des fonctionnalités principales étaient implémentées, nous avons fait le choix stratégique de les consolider. Nous les avons déjà mises à l'épreuve en les testant. Cela nous a permis de supprimer quelques bugs aussi bien au niveau du serveur, que du client et du driver pour Nao. Cette phase de test nous a aussi permis d'identifier quelles seront les fonctionnalités suivantes à implémenter.

Dans la suite du projet, nous allons essayer d'implémenter la dernière fonctionnalité essentielle manquante. Nous allons de plus rendre l'utilisation du serveur et du driver beaucoup plus accessible pour des novices. Nous avons déjà créé un script d'installation qui permet d'installer toutes les dépendances pour le projet. Mais comme ce projet est open source, nous souhaiterions que son utilisation soit accessible afin que des gens l'utilisent pour assister à des cours, pour donner des conférences, pour visiter un être cher ou encore pour explorer des environnements nouveaux.


Pour mettre en place la transmission d'image jusqu'au client Web, nous avons eu deux étapes importantes. La première a été de transmettre les images depuis le serveur vers le client. Ainsi, nous avons pu avoir des images aléatoires (de chats) qui défilent à 30fps sur le client web. Pour se faire, on utilise le moteur de rendu du navigateur. Le moteur de rendu n'attend pas la fin de la transmission de l'image pour l'afficher au client. Ainsi, nous pouvons envoyer un flux vidéo continu depuis le serveur vers le client.

Transmission d'images

Ensuite, il nous a fallu envoyer les images depuis Nao vers le serveur. La documentation de chez Aldebaran nous renvoyant vers trois proxys différents, nous avons décidé qu'il serait intéressant d'exposer les différentes options offertes au projet et de faire un choix commun. Alors le membre en charge de la transmission vidéo a préparé une soutenance en anglais présentant les trois proxys à toute l'équipe, en détaillant les enjeux et possibilités.

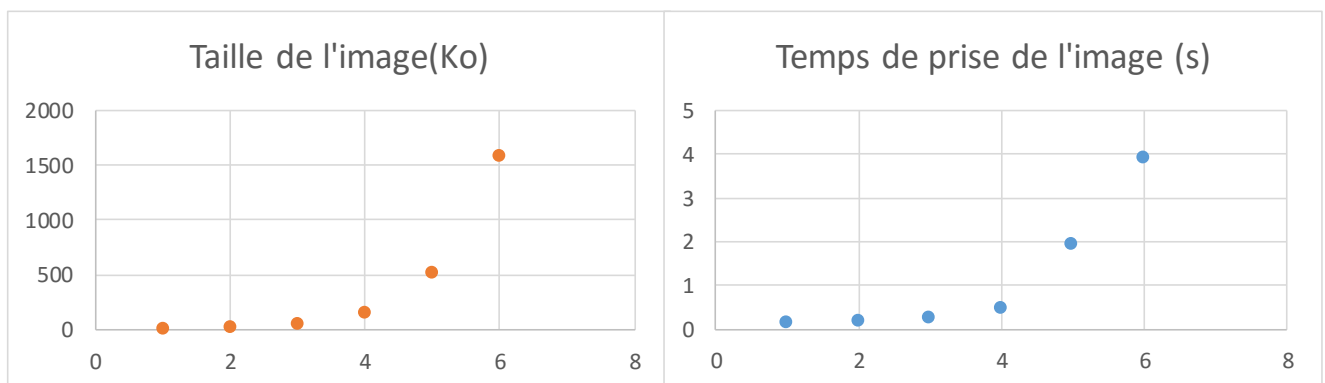
Summary

- What are the possibilities ?
- How it works ?
 - VideoRecording
 - VideoDevice
 - PhotoCapture
- Procedure



Après près de trois séances passées à développer une solution, les premières images prises par Nao ont pu s'afficher sur l'écran de l'ordinateur. Deux heures plus tard, il était possible de transmettre les images vers le client web. Nous avons alors testé plusieurs manières de transmettre les données. La plus simple a été de transmettre une image en format .png. Nous avons essayé d'utiliser les méthodes de la bibliothèque PIL, mais une erreur de version nous empêchait de travailler directement avec des méthodes de transmission de données brutes, sans spécifier de format.

Maintenant, nous ne savons quelle qualité de l'image choisir pour avoir un confort d'utilisation et ne pas trop surcharger le processeur et l'infrastructure réseau utilisée par Gnubiquity.



C'est alors qu'après plusieurs test chronométrés (avec la fonction `time()`), nous avons pu établir les graphiques suivant :

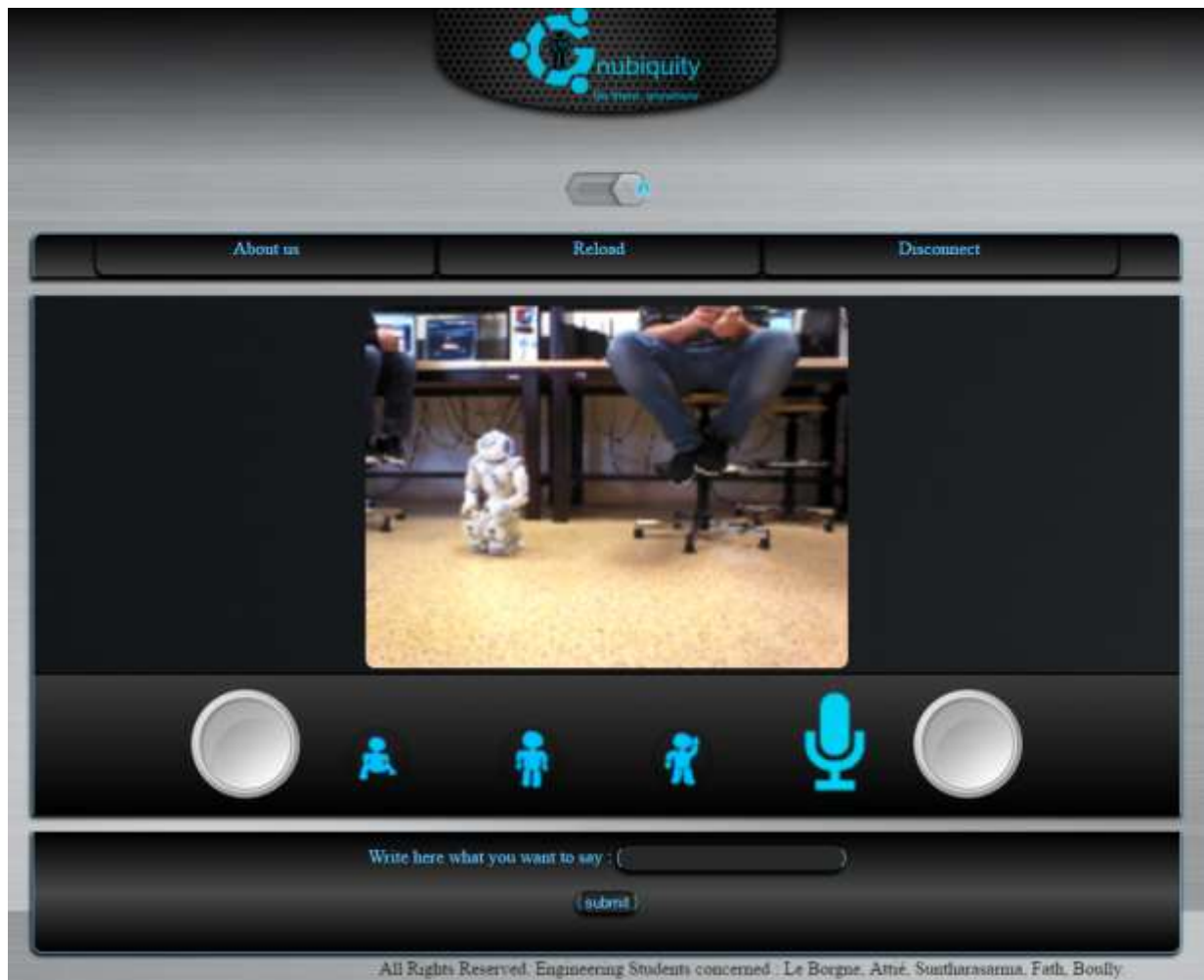
Lorsque nous croisons les deux graphiques, il apparaît que la taille de l'image croît de manière linéaire avec le temps qu'il faut pour prendre et afficher l'image sur l'ordinateur. Nous concluons alors que la connexion entre le robot et le serveur est ici le facteur limitant, au vu du format de l'image transmise.

Ainsi, nous pouvons porter une amélioration significative sur la transmission vidéo en mettant en place un système de compression plus performant ou en choisissant un robot avec une connexion plus performante que celle permise par Nao. D'autant plus que nous avons ici la chance de pouvoir utiliser la caméra de nao tout en le faisant bouger ou parler (grâce au proxy VideoDevice).

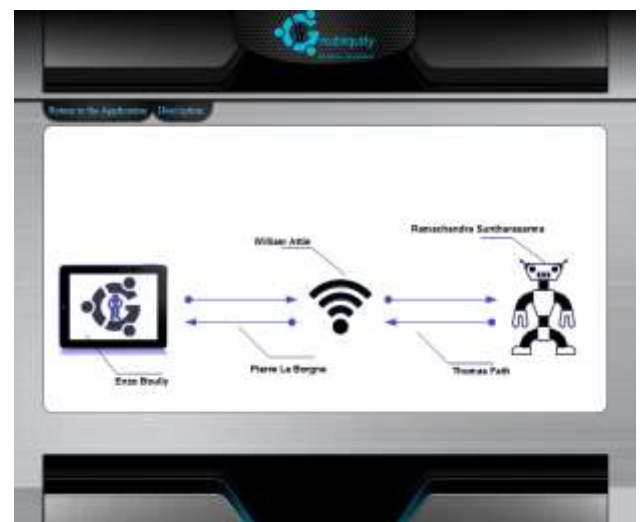
Développement de l'application Web :

L'application prend en compte trois pages :

La page principale :



L'onglet About us :



Description :



Dans un premier temps, nous avons établi la liste des fonctionnalités à implémenter pour le projet à savoir (par ordre décroissant d'importance) :

- Contrôle du robot
- Retour vidéo
- Retour audio
- Envoi audio
- Lecture de texte

De plus, L'application devait répondre à d'autres critères parmi lesquelles :

- l'ergonomie
- le design
- le caractère responsiv design
- la facilité de compréhension de l'application

Pour le contrôle du robot, deux joysticks ont été mis en place ainsi que des boutons de positions prédéfinies telles que : debout (prêt à l'activité), assis (repos), et lever le bras. De plus, il est possible de tourner la tête du Robot en touchant sur l'écran, la zone où l'on souhaite se centrer pour regarder.

Le retour vidéo a été réalisé de manière à ce que la qualité puisse être réduite au profit de la taille de l'image vidéo fixe. Ce choix s'explique notamment par le fait que nous avons le contrôle de la rotation bi-axiale de la tête du robot. De ce fait, il fallait une taille minimale du retour vidéo afin d'être un minimum précis lors du touché de l'écran.

L'envoi audio a été réalisé de deux manières distinctes :

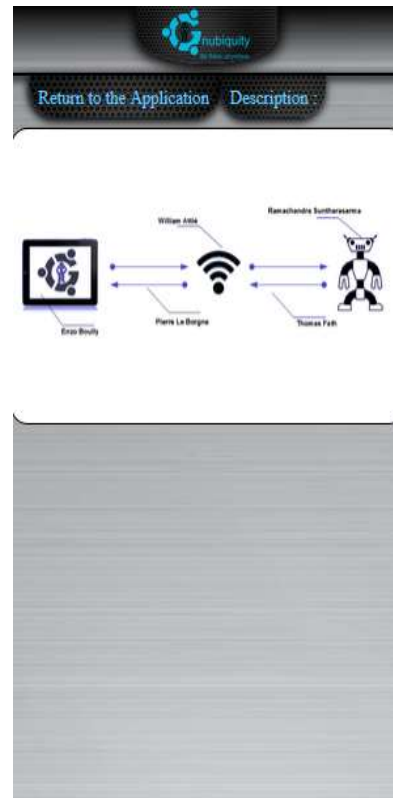
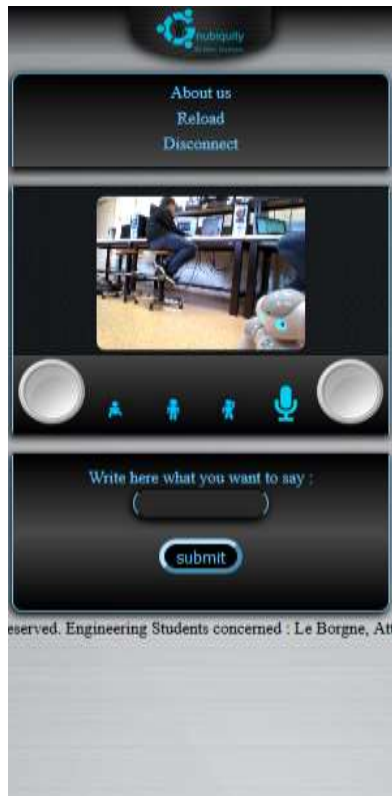
- Grâce à l'accès au micro dans un premier temps qui renvoi par conséquent notre propre voix,
OU
- Grâce à un champ texte qui permet d'écrire ce que le robot dira.

Remarque : Cette dernière fonctionnalité permet entre autres à l'éventuel malade de pouvoir quand même interagir avec l'entourage du Robot même s'il ne peut pas parler.

Ce sont des critères importants pour une lecture et compréhension de l'application irréprochable.

L'aspect CSS a été vraiment mis en avant avec un poids égal aux autres langages de programmation.

Il était nécessaire de pouvoir utiliser notre application que ce soit sur un téléphone, une tablette, ou un ordinateur d'où l'influence de l'aspect responsive Design. Nous voulions une application simple, intuitive et design :



Outils et méthodes de développement

Pour finir, il est important de préciser que nous avons utilisé un outil de développement en commun (GitHub) en tant que novices, sauf pour un membre de l'équipe.

C'est pourquoi tout le monde n'a pas forcément assis les bons réflexes de développement en commun. C'est la raison qui nous a obligés à passer une séance entière à revoir le code fonctionnalité après fonctionnalité afin de s'assurer du bon fonctionnement du projet global.

C'est alors que viens la dernière compétence technique acquise dans ce projet : celui de maîtriser le code que nous développons afin de savoir comment l'intégrer, le tester et le partager avec les autres collègues développeurs, en évitant de bloquer le projet dans son intégralité.

Une fois cette mise en commun, nous étions soulagés d'avoir enfin un aperçu fonctionnel et aboutis de nos trois semestres de labeur. Au final, il nous restait plus qu'à repousser l'application dans des situations contraignantes pour voir comment le système réagit et pouvoir porter les correctifs nécessaires.

C'est donc dans une séance ludique et amusante qu'il nous est apparu qu'il est important de gérer les conflits pour que le robot puisse rester fonctionnel malgré que plusieurs utilisateurs essaient de le manipuler.

Conclusion

Nous avons pu débuter ce semestre avec un projet où tous les choix techniques ont été testés et validés. Ainsi, au long des 14 séances de projet, nous avons pu implémenter les fonctionnalités les unes après les autres, sans mauvaise surprise. Grâce à une ambiance de travail agréable et une méthode de travail en groupe efficace, nous avons pu implémenter la majorité des fonctionnalités essentielles de Gnubiquity. Avec la méthode de développement Scrum-Agile, nous avons été capables de concevoir, développer et tester un produit tout en contournant les multiples limitations matérielles.

De plus, ce projet a permis à toute l'équipe de découvrir et de mettre en application de solides méthodes de travail alliant ambiance saine, outil de travail collaboratif de pointe et l'avancement d'un projet ambitieux, ludique et intéressant.

Durant le dernier semestre qu'il reste, l'équipe sera réduite à trois membres. Il nous reste à mettre en place quelques dernières fonctionnalités permettant une utilisation sécurisée des robots. Ensuite, nous allons communiquer autour de la solution développée.