POLYTECH ANNECY-CHAMBÉRY

UNIVERSITÉ SAVOIE MONT BLANC

APP
Robotique de service

Rapport semestre 6

BENHIMA Mehdi
JENNY Camille
LEGLISE Cloé
MISON Jules
ROY Nicolas
RUIZ Florian
SALH Hamza
SANGOUARD Marine

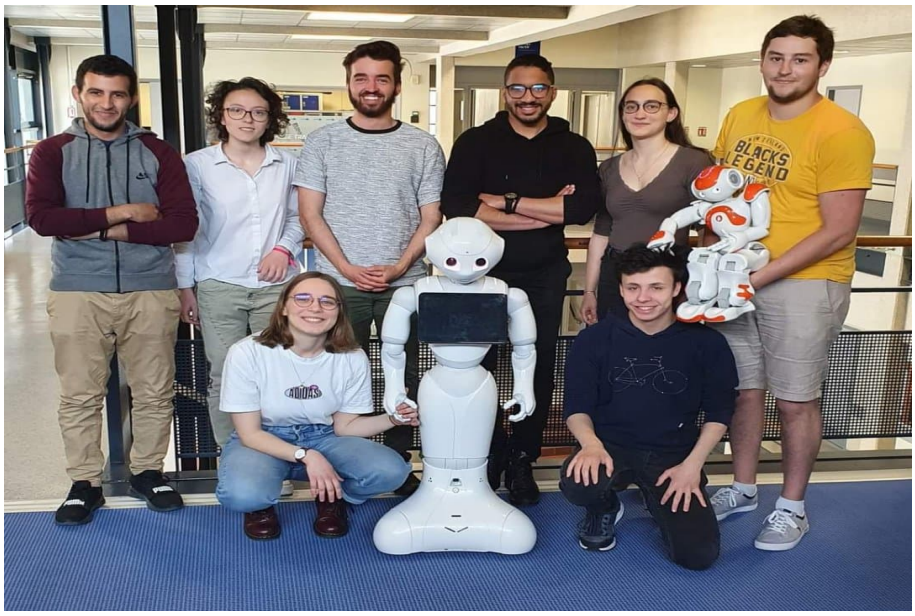

Table des matières

Introduction

Dans le cadre de notre formation d'ingénieur en filière « Systèmes Numériques et Instrumentation » (SNI), nous avons un apprentissage par projet à réaliser sur un thème au choix. Le nôtre est la robotique de service. Ce projet se déroule sur les trois années de notre formation, et l'objectif à court terme est défini par semestres ; l'objectif à long terme étant défini par l'équipe au début du projet. Ainsi premier semestre est dédié à la découverte des différents outils mis à notre disposition et à la définition de notre objectif à long terme pour le projet. Les semestres suivants sont respectivement axés sur la mise en place d'un cahier des charges, le développement du corps du projet, et la valorisation de notre travail.

Ainsi, ce premier semestre nous a servi à comprendre le fonctionnement des différents logiciels et robots, ainsi qu'à leur maîtrise, et à explorer les différents outils extérieurs dont nous aurons besoin. Ce travail nous a en réalité permis de commencer une ébauche de cahier des charges et de débuter significativement une la partie fonctionnelle du projet.

Objectifs

Le thème de notre projet est la robotique de service. Notre but est donc d'utiliser des robots à notre disposition, c'est-à-dire principalement des modèles de la société Aldebaran Pepper et Nao, pour subvenir à des besoins de services à l'être humain, par exemple dans un contexte de guidage. Ce qu'on appelle guidage est plus globalement la faculté du robot à guider des personnes dans un bâtiment quelconque, tout en interagissant avec elles, afin de leur indiquer leur chemin.

Pour commencer ce projet, nous avons choisi de se baser sur le bâtiment de Polytech Annecy et plus précisément de programmer Pepper pour qu'il puisse déjà se repérer dans le deuxième étage où notre salle d'APP se situe ; le but étant de parvenir à faire de lui un guide de l'école lors d'une journée portes ouvertes en indiquant les salles intéressantes à visiter pour les élèves. Pour ce faire, nous avons identifié différents objectifs spécifiques aux robots.

Pour ce qui est du robot Pepper, il faudrait que celui-ci soit capable de se repérer dans son environnement au sein d'un bâtiment, c'est-à-dire qu'il puisse identifier les salles autour de lui afin de proposer des trajets aux visiteurs, ainsi qu'éventuellement accompagner lesdits visiteurs à la destination de leur choix. Pour que cet accompagnement soit possible, il faut que nous réussissions à fluidifier au maximum le déplacement de Pepper pour qu'il se déplace en ligne droite, sans confondre les portes avec des couloirs, et ne se perde pas. Il faudrait également qu'il soit capable d'interagir avec les visiteurs, que ce soit à l'aide de la synthèse vocale ou en utilisant la tablette tactile intégrée au robot.

Le robot Nao n'étant pas très stable, nous avons imaginé pour lui un rôle immobile d'accueil des visiteurs, probablement sur une table dans le hall. Il pourrait ainsi apporter aux visiteurs une présentation générale et courte de l'école et des différentes filières, à la demande de l'utilisateur, ce qui implique une notion d'interaction par synthèse vocale afin de proposer ses services et réagir en fonction des réponses reçues. L'idéal serait également de mettre au point un système de communication entre les robots pour que Nao puisse prévenir Pepper lorsque des visiteurs souhaitent se rendre à l'étage et utiliser ses services.

Si nous n'avons pas pu en bénéficier lors de ce premier semestre de travail, un deuxième robot Pepper devrait être mis à notre disposition pour la suite de ce projet, ce qui pourrait nous permettre de faire présenter le premier et le deuxième étage, chacun par un robot Pepper, qui pourraient communiquer entre eux.

Le but final de notre projet serait d'avoir une structure évolutive afin que nos programmes puissent être utilisés dans d'autres bâtiment, en en récupérant le plan sur le réseau, et ainsi faire office de guide dans n'importe quel musée ou autre bâtiment. Pour cela, il faut que les instructions de déplacement des robots soient adaptatives.

Nous avons également à notre disposition d'autres robots que nous n'avons pas su comment intégrer à notre projet pour le moment.

Répartition des rôles et organisation

Notre équipe étant forte de huit personnes, et notre projet plutôt important, l'organisation est primordiale. Nous avons réparti les rôles de secrétaire, responsable matériel, et animateur, sur une base tournante dès la première séance afin que chacun connaisse son rôle avant même les séances. Pour ce qui est des différentes tâches à réaliser sur le projet en lui-même, chacun s'est naturellement dirigé vers ce qui lui plaisait le plus. Globalement, l'organisation n'a jamais posé problème, chacun était satisfait de sa tâche et nous n'avons pas eu à faire face à des imprévus majeurs ou des absences gênantes.

Florian Ruiz a été désigné chef de projet, c'est donc lui qui doit veiller à ce que le groupe reste sur la ligne de travail que nous avons définie ensemble et procéder à l'organisation de réunions régulières pour partager nos avancées et organiser la suite.

Concernant les tâches spécifiques au projet, nous avons effectué une réunion au début du projet pour découvrir quelles étaient les forces et faiblesses de chacun, ainsi que les compétences que chacun avait envie de travailler. En fonction de ces critères-là et de ce que chacun attendait du projet, nous nous sommes répartis en plusieurs équipes en fonction des différentes parties mises en évidence que constitue notre projet. Chacun a pu donc se focaliser sur un aspect du projet, qu'il a pu approfondir à chaque séance de ce semestre, tout en s'intéressant aussi à ce que les autres membres du groupe effectuaient. En effet, à la fin de chaque séance nous faisons un tour de table pour expliquer (plus ou moins précisément) ce qu'on avait fait pendant la séance.

Au fur et à mesure de notre avancement, nous avons différencié quatre axes de travail principaux : le déplacement du robot Pepper, la synthèse vocale pour l'interaction avec les deux robots Nao et Pepper, l'utilisation de la tablette tactile intégrée au robot Pepper, et enfin le serveur permettant de lier tous ces éléments en un projet cohérent.

Pour ce qui est de la répartition des membres dans les différents axes de travail, nous avons procédé de la manière qui suit.

Nicolas Roy a décidé de travailler sur le déplacement du robot Pepper car il avait déjà des bases concernant les algorithmes de plus court chemin et que la programmation de systèmes embarqués l'intéressait. Il voulait par conséquent développer ses compétences dans ce domaine.

Camille Jenny a également voulu se concentrer sur le déplacement de Pepper, puisque son appétence naturelle pour les robots est ce qui l'avait conduite en SNI en premier. Elle a ainsi pu découvrir plus en profondeur la programmation par langage graphique, mais aussi le langage Python, et mieux découvrir le système de fonctionnement du robot Pepper.

Jules Mison s'est concentré sur la partie synthèse vocale pour pouvoir améliorer ses compétences en programmation, notamment en Python. La synthèse vocale se travaillant sur le logiciel Aldebaran Choregraphe, cela lui a permis de découvrir et d'améliorer ses compétences dans ce langage de programmation. Le fait de pouvoir travailler sur des robots est une grande source de motivation.

Mehdi Benhima s'est concentré sur la partie de la création du serveur car il était intéressé de savoir comment la connexion s'effectuait entre les différents systèmes de notre projet. Cela lui permettrait aussi de développer ses compétences au niveau de l'interface client / serveur et sur les langages de programmation que sont le Python et le Java.

Hamza Salh s'est focalisé sur la partie tablette de Pepper, donc de l'interface homme / machine. Cela lui a permis de travailler ses compétences en programmation, notamment en

CSS, PHP, HTML et Javascript, pour réaliser une application web permettant d'envoyer des ordres et recevoir des informations via Pepper.

Cloé Léglise s'est intéressée à toutes les parties de notre projet, préférant ne pas se concentrer sur une tâche unique. La polyvalence de son rôle lui as permis de pouvoir rendre service à l'équipe qui avait le plus besoin d'aide à chaque séance, elle a ainsi passé beaucoup de temps avec l'équipe du déplacement de Pepper.

Marine Sangouard s'est focalisée sur la partie gestion de notre projet, notamment en tâchant de prendre en main GitHub, qui est un outil très utile pour la gestion de projet, centraliser et partager les différentes parties de notre projet. Dans le projet en lui-même, Marine ne s'est pas non plus concentrée sur une tâche en particulier, préférant pouvoir travailler toutes les compétences utiles au projet et avoir une vue d'ensemble du projet.

Florian Ruiz ne s'est pas focalisé sur une tâche en particulier, son souhait était de travailler sur les robots en général, donc il a passé la majeure partie de son temps à travailler sur Choregraphe, aussi bien s'intéressant au déplacement de Pepper qu'à la communication avec les deux robots. Tout cela dans l'optique d'améliorer ses compétences en programmation.

L'avantage d'être assez nombreux dans un groupe est que nous pouvons nous entraider plus facilement en utilisant toutes les compétences de chacun. Nous pouvons aussi avoir plusieurs points de vue et des regards de vérification différents sur tout ce que l'on fait. Cependant, les prises de décisions peuvent prendre plus de temps car les points de vue peuvent diverger beaucoup plus que lorsque l'on travaille avec moins de personnes. Nous devons alors apporter des arguments plus conséquents pour convaincre l'ensemble du groupe. De plus, plus d'idées viennent à nous, ce qui peut être utile lors de phase de créativité, mais moins lorsque nous devons concentrer nos idées et ne pas déborder du sujet par exemple. Ainsi, l'organisation et la communication doivent être les plus précises possibles pour que le projet avance rapidement, ce qui n'est pas toujours simple.

Base du projet et point de départ

Lors de la mise en place de notre projet et de la définition de nos objectifs, nous avons pu bénéficier des conseils de nos professeurs, mais nous avons également bénéficié des travaux des élèves plus anciens ayant déjà travaillé sur des projets similaires via l'APP. En-dehors des conseils qu'ils ont pu nous donner, nous avons également bénéficié d'une partie de leur travail, ce qui nous a permis d'avancer plus rapidement dans nos recherches, mais également de définir ce qui était ou non réalisable dans nos différentes idées. Nous avons bénéficié d'une partie de leur programme sur le déplacement du robot Pepper, la création d'un serveur, et la communication avec un client. Nous avons donc pu nous baser sur ces programmes pour mieux comprendre le fonctionnement de ces différentes fonctionnalités et essayé de faire de nos propres programmes quelque chose d'encore plus robuste et poussé puisque nous n'avions pas à nous occuper de trouver les bases par nous-mêmes.

Nous avons ensuite réfléchi à la dimension de notre projet, c'est-à-dire toutes les possibilités de développement et de réalisation que nous pouvions amener à terme tout en prenant en compte les limites et difficultés que nous pourrions rencontrer. Nous avons donc mis au jour les différentes problématiques sur lesquelles travailler, c'est-à-dire ce qui a été précisé plus haut, et leur compatibilité lors de la mise en commun via le serveur.

Travail effectué durant le semestre

La synthèse de ce que nous avons réussi à mettre en place jusqu'à présent sera découpée selon les quatre axes de travail ayant déjà été mentionnés.

Déplacement du robot Pepper

Suivi de mur et résultats

Le déplacement du (ou des) robot Pepper est une composante critique du projet considérant l'objectif envisagé. Durant la période de prise en main du projet, un effort a été fait afin d'établir les possibilités et l'étendue du projet relativement au déplacement des robots.

En premier lieu, l'avancement et les solutions proposées par les équipes précédentes ayant travaillé sur des projets similaires ont été consultés afin d'établir un point de départ. Afin de pallier aux difficultés rencontrées dans les projets précédents lors des déplacements, les codes sources ont simplement servi d'inspiration et un programme original est établi. L'une de ces difficultés est notamment l'architecture de déplacement basée sur une boucle de contrôle ouverte. Cette architecture entraînait des erreurs considérables sur les distances parcourues (par rapport à la distance commandée) et une difficulté au suivi de mur.

Un nouveau bloc de déplacement a été mis en place. Ce dernier a pour fonction de permettre à un robot Pepper de se déplacer d'une distance x, longeant une surface, tel un mur, à sa droite ou à sa gauche. (Un déplacement non guidé a aussi été implémenté.) Ce bloc original utilisable sur la plateforme de programmation Choregraphe comprend trois paramètres principaux : la distance commandée [m], la distance du mur et la référence de suivi [surface à droite, surface à gauche ou déplacement libre].

L'architecture de ce bloc est centrée sur une boucle fermée contrôlant la vitesse et l'angle de déplacement du robot. Lors de chaque boucle, l'angle de déplacement du robot (voir figure 1) relativement à la surface cible sera calculé et corrigé. Traditionnellement, la distance parcourue depuis le départ du mouvement est calculée en utilisant les capteurs de vitesse à élément magnéto résistif du robot. Cette valeur est utilisée en comparaison avec la distance commandée afin de mettre fin au déplacement lorsque nécessaire. Finalement, la distance relative à la surface cible est mesurée afin de garder le robot à une distance relativement constante de la surface ou centrer le robot si deux surfaces sont détectées. Il faut noter que si une surface est détectée trop près du robot, le mouvement vers l'avant cessera le temps que le robot s'éloigne du mur avant de continuer son trajet. Cela permet d'éviter les collisions.

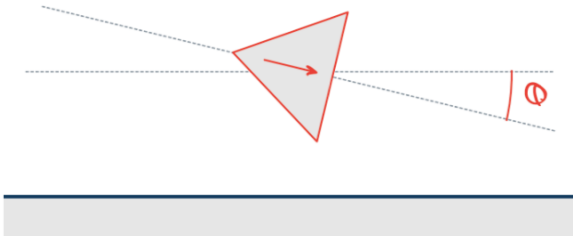

Figure 1 : Angle de déplacement (θ) à corriger

Les essais pratiques du déplacement montrent des résultats positifs au niveau de la capacité de suivi de mur et d'erreur sur la distance parcourue. Le tableau suivant montre l'erreur sur certains déplacements réalisés :

No	Distance commandée [m]	Distance parcourue [m]	Erreur [%]
1	12.5	12.39	0.88
2	12.5	12.35	1.2
3	7	6.95	0.74

Figure 2 : Erreur sur la distance parcourue

Problèmes rencontrés et solutions

Le développement du bloc de déplacement sera poursuivi durant les prochains semestres d'APP.

Quelques difficultés ont été rencontrées lors du développement du bloc permettant un suivi de mur. Premièrement, la commande de déplacement « *ALMotionProxy::moveTo* » ne permet pas un déplacement consistant et présente une erreur importante sur la distance parcourue par rapport à la distance commandée. De plus, un déplacement fluide d'un suivi de mur est impossible. Pour pallier à ce problème, nous avons mis en place une architecture en boucle fermée, basée autour de la fonction « *ALMotionProxy::moveToward* » qui permet un déplacement fluide.

Deuxièmement, les distances mesurées par les capteurs laser dont le robot Pepper est équipé peuvent fluctuer considérablement entre chaque lecture. Il est important de prendre en compte ces fluctuations possibles. L'implémentation de seuils et d'une boucle fermée beaucoup plus rapide que le mouvement mécanique du robot permet de pallier aux fluctuations de mesure qui affecte directement la correction de la boucle fermée.

Enfin, le programme interne du robot Pepper prend en charge par défaut l'anticollision du robot. Bien que cela simplifie beaucoup la programmation d'un point de vue de la sécurité, il empêche souvent un déplacement fluide du robot puisque le périmètre de détection d'obstacle est très sensible et relativement grand. Par exemple, si une personne croise le robot Pepper dans un couloir, le robot Pepper cesse automatiquement tout mouvement. Cette situation ne devrait pas causer un arrêt complet du déplacement du robot. Le système interne d'anticollision peut être outrepassé par l'utilisateur. En revanche, cela nécessite l'implémentation d'un système d'anticollision par l'utilisateur. Ceci n'est pas encore mis en place.

Algorithme de gestion de parcours

Bien que pas encore implémentée, une conception haut niveau de l'architecture de guidage du robot a été établie.

Afin de permettre un déplacement efficace du robot Pepper dans les couloirs d'un bâtiment, un graphe et un algorithme du plus court chemin seront utilisés. Ceci, en combinaison avec les différents blocs Choregraphe créés devrait permettre le déplacement du robot à l'intérieur de n'importe quel bâtiment, pourvu qu'un graphe adéquat soit disponible.

Un graphe adéquat permettant le déplacement d'un Robot Pepper dans un bâtiment représente l'ensemble des chemins possibles que le robot peut emprunter. En d'autres mots, ce sera une représentation des chemins possibles du bâtiment, interprétable par l'algorithme de plus court chemin. Il sera caractérisé par, au minimum, un sommet à chaque intersection de couloirs du bâtiment et un sommet à chaque destination clé choisie.

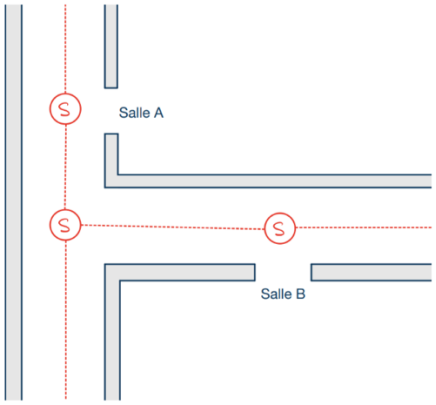

Figure 3 : Emplacement envisagé des sommets du graphe

Chacun des sommets contiendra au minimum les informations suivantes : un identifiant de type chaîne de caractères, qui sera l'identifiant unique du sommet ; des coordonnées (x,y) en mètres, de type couple de flottants, représentant l'emplacement du sommet relativement à un point de départ constant ; et éventuellement la présence de Naomark, de type booléen permettant de savoir si une Naomark (qui permet la recalibration de la position de Pepper) se situe à ce sommet.

Les arêtes entre les différents sommets représenteront l'ensemble des chemins linéaires sur lesquels un robot peut se déplacer. Chacune des arêtes doit donc représenter un mouvement possible non obstrué en ligne droite. Une arête reliant deux sommets sera caractérisée par la distance à parcourir entre ces deux sommets et un indicateur permettant de déterminer si un suivi de mur est nécessaire ; si oui, on indiquera si le mur droit ou gauche doit être suivi.

L'algorithme de Floyd Warshall sera utilisé pour déterminer le plus court chemin entre deux points du graphe, correspondant au point de départ du robot et la destination voulue dans le bâtiment. Cet algorithme permettra de calculer préalablement au déplacement les plus courts chemins de tous les sommets vers tous les sommets selon le poids (distance à parcourir) de chaque arête. Ce calcul sera fort probablement réalisé par le serveur considérant la capacité de calcul limitée des robots Pepper. Lorsqu'un déplacement est nécessaire, une chaîne de sommets à parcourir pourra être générée et ainsi guider le robot vers la destination.

La gestion de l'orientation du robot sera particulièrement importante durant ses déplacements puisqu'il doit connaître la direction à suivre afin de rejoindre le prochain sommet du graphe. Une gestion stricte de l'orientation du robot permettra de facilement négocier les intersections du bâtiment. En considérant les coordonnées du dernier sommet visité par le robot Pepper (provenance), du sommet sur lequel le robot Pepper se trouve (position actuelle), et du prochain sommet (destination), il est possible de calculer les rotations nécessaires pour atteindre le sommet cible. La démarche qui suit peut être utilisée pour déterminer la rotation nécessaire.

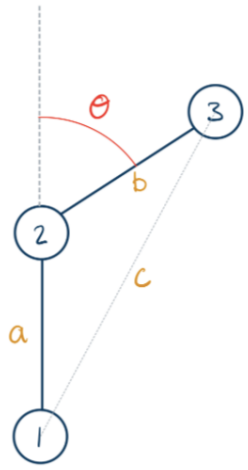

Figure 4 : Angle de rotation pour atteindre le prochain sommet

$$\theta = \pi - \cos^{-1}\left(\frac{a^2+b^2-c^2}{2ab}\right)$$

(Les distances a, b et c peuvent être calculées à partir des coordonnées du sommet.)

Par la suite, un bloc Choregraphe de rotation peut être utilisé pour orienter le robot.

Perspectives

Afin d'atteindre les objectifs définis lors de la prise en main de l'APP, le développement de certains aspects du projet doit être réalisé. La section suivante détaille ces perspectives. Ces points peuvent être interprétés comme les lignes directrices du travail à réaliser durant les prochaines sessions d'APP.

L'équipe doit créer un bloc ou utiliser un bloc Choregraphe déjà existant qui permet une simple rotation du robot sur lui-même d'un angle donné. Ce bloc sera utilisé en combinaison avec celui créé lors du semestre (bloc de déplacement) afin de permettre à un robot Pepper de se rendre du point A au point B dans un bâtiment. Ses déplacements seront composés d'une séquence de suivi de mur et de rotation aux intersections.

Il sera nécessaire de définir et mettre en place la communication entre le serveur et le robot Pepper. Lorsque demandé, le serveur fournira au robot Pepper une séquence d'actions détaillée à suivre afin qu'il atteigne la destination voulue. Le serveur sera donc responsable du calcul du plus court chemin et de la traduction de la séquence de sommets obtenue par l'algorithme du plus court chemin en une séquence d'action à exécuter par le robot Pepper (séquence de suivis de mur, de déplacements à l'aveugle et de rotations).

Bien que le bloc de déplacement créé durant la session permette au robot de se déplacer relativement bien, certains scénarii problématiques ne sont pas encore gérés et peuvent empêcher le robot d'atteindre la destination voulue. Ces scénarii connus à présent sont les suivants : le robot Pepper doit traverser une porte ouverte ou un autre type d'ouverture dans le mur utilisé comme référence pour donner l'angle au robot ; le robot Pepper doit contourner un obstacle inattendu sur son parcours.

L'utilisation de Naomark permettant la calibration de la position du robot Pepper serait très bénéfique pour augmenter la précision de la position et donc les chances que le robot Pepper atteigne bien la destination voulue. Pour ce faire, le robot Pepper doit être capable d'identifier une Naomark, de la lire et d'en envoyer les données vers le serveur.

Communication des robots

Dans cette partie, qui abordera la communication entre les robots et les clients, les schémas présenteront un fonctionnement simplifié pour une meilleure compréhension du lecteur.

Lecture de texte et enregistrement

Étant donné que nous voulons créer des robots qui peuvent s'adapter facilement à leur environnement, nous avons créé un bloc qui permet de facilement changer ce que dit le robot. Pour cela, nous avons créé un bloc qui prend en entrée un fichier texte et qui le fait lire par le robot (grâce au bloc Say Text).

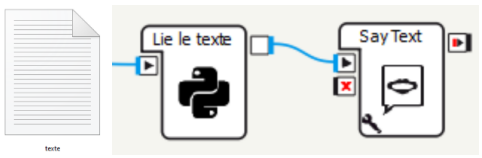

Figure 5 : Schéma de fonctionnement de la lecture de texte

Nous avons également créé une variante de ce bloc qui permet au robot de lire une ligne précise du document. Pour cela, le bloc avait en entrée le texte et le numéro de la ligne à faire lire au robot.

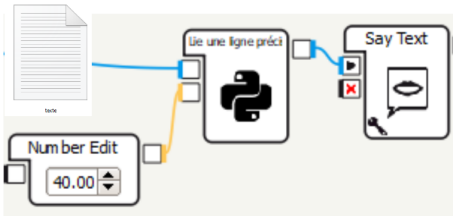

Figure 6 : Schéma de fonctionnement de la lecture d'une ligne spécifique

De même que pour la lecture d'un fichier, nous avons voulu faire en sorte que le robot soit capable d'écrire un fichier texte afin de l'envoyer au serveur. Ce bloc prend en entrée le lien du fichier et le texte à mettre dedans.

Figure 7 : Schéma de fonctionnement d'écriture dans un fichier texte

Reconnaissance vocale

Pour la reconnaissance vocale, nous avons remarqué qu'il existe un bloc « Speech Reco » qui arrive à reconnaître un mot parmi une liste possible de mots.

Figure 8 : Word list utilisée par le speech reco

Nous avons donc modifié ce bloc afin de lui permettre de reconnaître un maximum de mots. Pour cela nous avons créé un fichier texte avec une liste des 700 mots les plus utilisés dans la langue française en ajoutant des mots importants dans notre utilisation (Nao, Pepper, C210…). Puis nous avons repris notre bloc de lecture de fichier afin de donner ce fichier texte comme Word list. Cette technique a un gros inconvénient si on veut pouvoir parler normalement (en faisant des phrases). En effet avec cette solution le robot est capable de comprendre un seul mot à la fois. Donc pour lui dire une phrase on est obligés de lui dire mot par mot ce qui manque de fluidité et rend la conversation peu naturelle.

Nous avons donc réfléchi à une autre solution : utiliser le micro du robot pour isoler la phrase de son interlocuteur et l'enregistrer dans un fichier son. Il sera par la suite envoyé au serveur pour être traité. Dans le logiciel, il existe un bloc « Record Sound » qui permet d'enregistrer le son jusqu'à interruption du bloc. On ajoute un système de détection de fin de phrase (Grâce au bloc « Sound Loc » on met un minuteur d'une seconde qui se réinitialise s'il entend un son). Cette méthode a un avantage, c'est qu'elle ne nécessite plus de parler mot par mot comme la méthode précédente. Cependant, le traitement n'est pas fait automatiquement, le robot crée simplement un fichier son de la phrase. Ce sera par la suite au serveur de traiter le fichier son pour en ressortir un texte à interpréter.

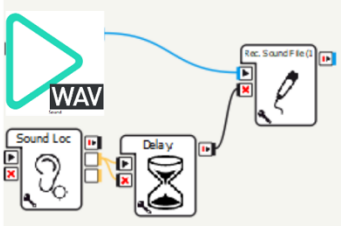

Figure 9 : Schéma de fonctionnement de la communication au serveur d'un fichier son

Difficultés rencontrées

Pour l'écriture d'un fichier texte par Nao, nous nous sommes rendu compte qu'à l'ouverture dudit fichier, celui-ci était vide. Nous avons par la suite compris que cela était dû au fait que le robot enregistre le fichier dans sa mémoire interne, à laquelle nous n'avons pas accès. Nous nous sommes alors servis du bloc de lecture du fichier par le robot pour confirmer que l'enregistrement fonctionnait correctement. Le même problème s'est posé pour l'enregistrement du fichier son, et nous avons utilisé une solution similaire via le bloc « Play Sound File ».

Pour créer notre Word list, nous nous sommes procuré une liste existante avec 700 mots. Cependant, le logiciel n'arrivait pas à la lire à cause de son caractère spécial. Nous avons donc dû remplacer les caractères spéciaux par leur caractère classique (é -> e, ç -> c, œ -> oe …).

Les essais ont été réalisés avec le robot Nao, dont la qualité des micros laisse à désirer et provoque beaucoup de bruit parasite dans les fichiers sons.

Perspectives

Pour poursuivre le développement de notre projet, il nous faut maintenant relier la communication robot au serveur. Pour ça il va falloir créer plusieurs blocs : un bloc de réception de données au niveau du serveur, et un bloc d'émission vers le serveur au niveau du robot.

Ensuite, il sera nécessaire de traiter les données reçues par le serveur. Si on lui envoie un fichier texte, il va devoir traduire les informations contenues et en déduire une action à effectuer. Il devra être capable de traiter des fichiers son, déterminer ce qui est dit et pouvoir comprendre la question qui est posée pour prendre une décision. Il est possible aussi qu'il y ait des fichiers son qui sont des « erreurs », étant donné que Nao a des micros de mauvaise qualité, il est possible qu'il enregistre un fichier son où personne ne parle au robot. C'est pourquoi le traitement de ce fichier par le serveur devra être capable de comprendre que ce fichier est « vide » ou ne concerne pas le robot. Pour faire cela, nous pourrons utiliser des algorithmes de reconnaissance vocale et de compréhension de texte.

Par exemple si le robot Pepper entend la phrase : « J'aimerais aller dans la salle C210 », le robot l'enregistre dans un fichier son, l'envoie au serveur, le serveur doit traduire le fichier son en texte, puis le comprendre pour envoyer l'information au robot de se déplacer vers cette salle.

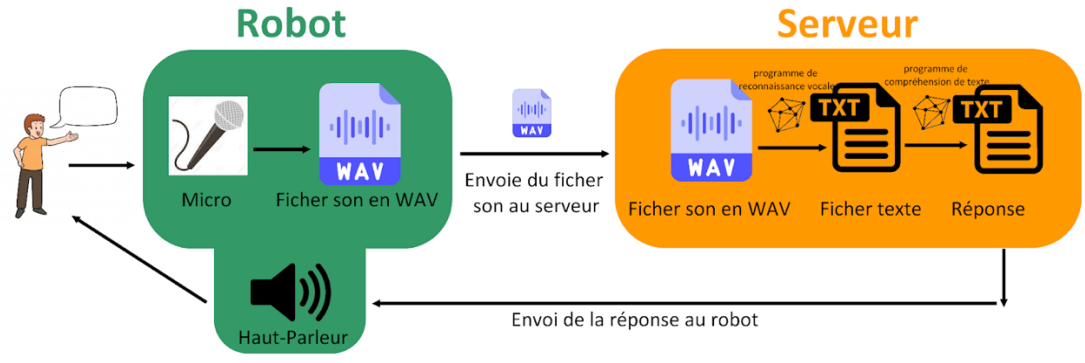

Figure 10 : Schéma de fonctionnement de la communication entre le robot et le serveur

Application web

Dans notre projet, nous avons plusieurs méthodes de communication Pepper. L'une d'entre elles utilise la tablette tactile intégrée au robot. Nous avons décidé de mettre en place

une application web permettant à l'utilisateur d'interagir avec Pepper en sélectionnant une destination au sein de notre école.

Objectifs

Nous avons défini différents objectifs pour cette application Web.

Premièrement, un bon accueil pour les visiteurs. En effet, la page d'accueil du site affichée doit être plaisante, simple d'utilisation et permettre de passer dans différents cas de figure en fonction de ce que l'utilisateur va choisir de faire.

Deuxièmement, une communication claire. La tablette tactile permet d'interagir avec le robot quand on lui demande son aide si jamais la synthèse vocale n'est pas suffisante (par exemple si le robot n'a pas compris la demande et qu'il ne peut donc pas réagir en conséquence). L'utilisateur a donc la possibilité à travers cette tablette de cliquer sur des choix sur la tablette au lieu de répéter à l'oral pour éviter à la synthèse vocale d'être perturbée par le bruit ambiant.

Troisièmement, l'affichage d'un plan. La tablette affiche le plan de l'étage et indique au visiteur comment aller à la destination choisie, en plus de proposer d'accompagner la personne. Nous avons pensé à la possibilité de faire scanner un QR code aux visiteurs pour accéder à ce plan directement sur leur smartphone, ou en entrant leur numéro pour qu'il soit envoyé par WhatsApp.

Quatrièmement, une traduction en anglais. Comme nous savons que Pepper est capable de comprendre et parler plusieurs langues, nous essayerons de laisser la possibilité de choisir la langue en appuyant sur des drapeaux que le site web proposera, si nous avons suffisamment de temps pour réaliser ladite traduction du site.

Enfin, nous avions envisagé la possibilité de rendre notre projet accessible aux personnes malentendantes avec un système de retranscription oral / écrit et une entrée de texte, mais ce dernier objectif a été abandonné pour des raisons de temps et de difficulté.

Avancement de l'application web

En faisant des recherches et à l'aide d'exemples des anciens étudiants, nous avons créé une première page d'accueil qui salue les visiteurs et leur demande de choisir une destination. Les différentes destinations possibles sont séparées en trois choix précis (salles de TP, salles de TD, amphithéâtre) pour accéder à ces locaux. Nous avons ainsi créé la page d'accueil suivante :

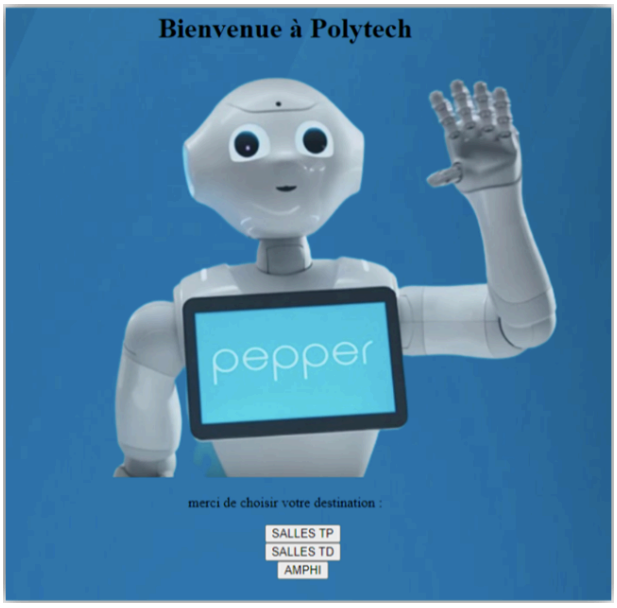

Figure 11 : Accueil de notre application web

Après que le visiteur ait effectué son choix à l'aide des différents boutons, il est renvoyé vers un autre lien contenant plus d'informations sur le choix qu'il a fait. La plan du deuxième étage est alors affiché, et l'utilisateur peut cliquer sur la salle de son choix afin de choisir de s'y rendre.

Figure 12 : Page du choix des salles de TP

```
<ul id="etage2">
    <li id="c213"><a href="?page=1&salle=C213"><span>c213</span></a></li>
    <li id="c214"><a href="?page=1&salle=C214"><span>c214</span></a></li>
    <li id="c215"><a href="?page=1&salle=C215"><span>c215</span></a></li>
    <li id="c218"><a href="?page=1&salle=C218"><span>c218</span></a></li>
    <li id="c217"><a href="?page=1&salle=C217"><span>c217</span></a></li>
    <li id="c216"><a href="?page=1&salle=C216"><span>c216</span></a></li>

</ul>
```

Figure 13 : Code HTML avec le lien des salles

De la même façon nous avons réalisé la destination vers les salles de TD et amphithéâtres. Nous avons ensuite ajouté un menu général du site permettant de circuler entre les différents éléments « accueil », « à propos de nous » et « help ».

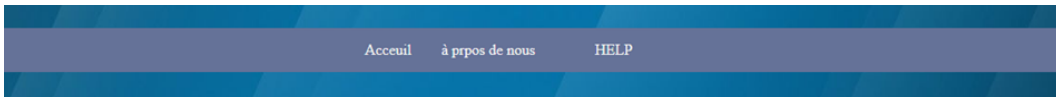

Figure 14 : Menu général du site

Dans le but de fermer le site et terminer toutes les informations nécessaires aux guide et utilisation nous avons accédé à une balise <footer> qui permet de remplir et traiter le pied de site.

Notre première version de site est totalement indépendante d'un serveur. L'interaction se fait seulement entre l'homme et la tablette de Pepper. Nous sommes actuellement en train de travailler pour intégrer cette version à un serveur ce qui nous permettra de commander Pepper.

Difficultés rencontrées

Nous avons rencontré plusieurs problèmes sur cette partie lors de ce semestre. Tout d'abord nous avons téléversé les pages et essayé de visualiser le contenu sur la tablette à travers Choregraphe mais il ne s'affiche pas correctement et pour d'autres parties les mesures faites ne sont pas convenables.

Perspectives

Pendant ce semestre nous sommes arrivés à faire un modèle de site normal avec des techniques classiques, par contre nous voulons résoudre les problèmes concernant la communication entre serveur et tablette à travers Choregraphe puis travailler plus sur les interfaces graphiques qui nous aident à faciliter la communication entre l'homme et machine, par exemple l'utilisation de Bootstrap ou PurCss. Nous souhaiterions également nous

améliorer en langage JavaScript afin de rendre le site plus dynamique et apprendre à utiliser des interfaces graphiques.

Serveur

Travail réalisé

Étant donné les différentes parties qui couvrent notre projet, nous avons conclu la nécessité de construire un serveur qui va nous permettre de connecter nos différents systèmes (Pepper, Nao, tablette de Pepper…), et c'est grâce à ce serveur que nous pourrons gérer les différentes parties de notre projet.

Au début du semestre, nous n'avions aucun pré-requis pour créer notre serveur, nous avons donc commencé par apprendre les bases dans ce domaine.

Nous commencerons par définir le terme « Client-Serveur ». En effet, c'est un protocole qui désigne un mode de transaction entre plusieurs programmes ou processus, l'un (ou plusieurs) qualifié de client qui envoie des requêtes au serveur, et l'autre qualifié de serveur qui attend les requêtes du client et y répond.

Nous avons cherché les différentes protocoles de serveur pour pouvoir bien choisir le protocole approprié à notre projet, et nous avons décidé de travailler avec le protocole TCP/IP, qui permettra la communication entre nos différents système grâce à leur adresse IP, de plus ce protocole présente plusieurs avantages tels que la possibilité d'établir une connexion entre différents types d'ordinateurs, un fonctionnement indépendant du système d'exploitation, la prise en charge de nombreux protocoles de routage, ou encore une architecture client-serveur très évolutive.

Nous avons ensuite essayé de créer notre premier serveur basé sur la formation de Monsieur Monnet en utilisant le langage Python comme langage support, après on a créé notre premier client en Python et nous les avons connectés et fait échanger des requêtes. Lorsque nous avons essayé de connecter plusieurs clients en même temps, le serveur ne répond plus comme nous le souhaiterions car notre serveur initial était un serveur one-thread, d'où nous nous sommes rendu compte de l'importance d'un serveur multi-thread vu qu'on vise à connecter plusieurs systèmes en même temps dans notre projet.

Le serveur multi-thread est un serveur ayant plusieurs threads, c'est-à-dire lorsqu'un client envoie la demande, un fil (thread en anglais) est généré à travers lequel un utilisateur peut communiquer avec le serveur. Nous devons donc générer plusieurs threads pour accepter plusieurs demandes de plusieurs clients en même temps.

On a donc créé notre serveur multi-thread tout en gardant le langage Python comme langage support, et nous avons créé nos clients et nous les avons fait communiquer avec le serveur en même temps, et le serveur est mieux parvenu cette fois à gérer les différentes requêtes en même temps.

La réponse du serveur était bonne, mais en termes de performance elle était limitée et sa réponse n'était pas trop satisfaisante vu le caractère single-thread de Python. Après plus de recherches sur le caractère robuste du serveur multi-thread de Java, nous avons décidé de prendre Java comme langage support pour notre serveur principal, et de coder les différents clients sur le langage le plus associé à chaque client (Python/Aldebaran pour Nao et Pepper et PHP pour la tablette).

Difficultés rencontrées

Lors de la communication du client avec le serveur, le serveur se bloque souvent sur le décodage du message envoyé par le client, et donc il fallait à chaque fois indiquer le codage utilisé par le client pour pouvoir l'utiliser pour décoder son message.

Nous avions du mal à connecter notre serveur Python avec un client extérieur en ayant le message d'erreur suivant : « Aucune connexion n'a pu être établie car l'ordinateur cible l'a expressément refusée », ce qui a également participé à notre décision de passer en Java.

Perspectives

Durant ce semestre, nous avons pris en main le fonctionnement du client serveur et ses différentes fonctionnalités, et nous avons testé le fonctionnement de notre serveur multi-thread avec des clients internes (sur la même machine), et nous avons réussi à le faire fonctionner. Pour le prochain semestre nous passerons à la prochaine étape, et nous connecterons nos différents clients (robots, tablette) avec notre serveur et nous allons les faire communiquer et échanger les différentes requêtes pour assurer la bonne gestion de notre projet.

Conclusion

Pour conclure, nous avons bien tous pris en main le matériel à disposition pour notre projet tels que les robots Pepper et Nao ainsi que le logiciel Choregraphe pour les programmer. Grâce à de nombreux tests, nous avons pris connaissance de tous les capteurs que possèdent ces robots en plus de toutes leurs capacités tels que le déplacement intelligent, l'interaction avec son environnement et la communication avec un serveur et donc avec d'autres entités.

De plus, nous avons pu avancer sur les principales parties distinguées grâce à nos recherches. Le déplacement du robot Pepper est en cours d'étude. La partie serveur web avec l'application web à afficher sur la tablette a pris plus de temps à se mettre en route car nous n'avions pas toutes les connaissances nécessaires dès le début du semestre. L'étude de la synthèse vocale est en cours de traitement également et la partie développement du serveur est encore une partie à maîtriser car aucun de nous n'a de connaissance sur le sujet.

Durant les prochains semestres, l'objectif sera donc de se baser sur tous nos apprentissages et avancées techniques pour mettre en place notre projet de robot guide. En effet, nous connaissons désormais différentes complications auxquelles nous pourrons faire face, ainsi que comment en surmonter certaines. Lors du semestre suivant, nous pourrons donc réaliser notre cahier des charges en connaissance de cause. Ainsi, grâce à ce semestre d'expérimentation, nous avons pu avancer dans notre projet dont l'objectif final serait d'utiliser Pepper et Nao en tant que guides qui pourraient être mis à disposition de n'importe quel bâtiment à l'aide d'un serveur et d'une carte des locaux. Le but de ce projet est d'être évolutif, et adaptable à tout type de local. De plus, l'université fera très probablement l'acquisition d'un deuxième robot Pepper d'ici à la fin de ce projet, ce qui signifierait un guide de plus qui fonctionnerait à partir du même programme et qui pourrait facilement être ajouté à notre serveur.