Rapport technique

22 mai 2022

Version 2

DIALOGUE ENGAGEANT AVEC QTROBOT (projet 04)

Auteurs (élèves de IMT Atlantique) : Maissa BEJI, Gabriel CHASTANET, Imane OKACHA, Victoria ANDRE

Destinataires : encadrant du projet Christophe LOHR et le COPIL (Comité de pilotage)

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Résumé

Ce projet a pour objectif de programmer le robot humanoïde QTrobot afin de lui permettre d'interagir avec les utilisateurs grâce à certaines conversations et des mouvements de la tête et des bras. Pour y parvenir, une approche de programmation basée sur le langage Python a été utilisée pour aborder ce sujet sous trois aspects.

Tout d'abord, le système de geste a été développé. Lorsque QTrobot prend la parole, il effectue maintenant des mouvements des bras et de la tête. Ces mouvements ont été conçus pour être le plus proche possible de ceux des humains lorsqu'ils parlent. Ils peuvent avoir plusieurs intensité ou être de différents types afin de reproduire également la diversité des mouvements chez les hommes (qui peuvent varier en fonction de la personnalité, de la nationalité…). Un deuxième aspect du projet a été d'utiliser les microphones du robot pour détecter la position de l'interlocuteur afin que le robot tourne la tête vers celui-ci lorsqu'il parle.

Pour terminer, afin de mettre en pratique les deux aspects précédents dans un cas plus concret, une fonction a été développée pour permettre à un humain de rechercher des informations sur Wikipedia. Le robot utilise des API pour interroger Wikipédia et ensuite lire à l'interlocuteur une partie du résumé correspondant à la recherche.

En conclusion, ce projet a permis de créer un système d'interaction avec QTrobot, capable de répondre à une recherche Wikipédia tout en faisant des gestes de la tête et des bras et en suivant l'interlocuteur du regard. Le dialogue expressif étant un besoin général pour l'interaction humain-robot, ce projet permet de continuer dans l'amélioration de ces interactions.

.

Table des matières

Introduction

Le projet se situe dans le domaine de l'interaction humain-robot et se concentre sur l'amélioration des fonctionnalités des robots humanoïdes pour les rendre plus engageants et plus naturels pour les humains. Plus spécifiquement, ce projet porte sur le robot humanoïde QTrobot [1, 2].

Actuellement, lorsque QTrobot parle ou bouge, il manque de vivacité et le comportement du robot peut sembler rigide et stéréotypé.

Le projet vise alors à exploiter les fonctionnalités avancées de la suite logicielle ROS* [3] (Robot Operating System) embarquée dans QTrobot. Cette suite logicielle offre la possibilité de programmer des comportements plus sophistiqués et autonomes en Python ou en C++.

La problématique est de rendre les interactions de robot aussi humaines que possible en utilisant les fonctionnalités ROS. Cela s'est fait selon trois tâches clés qui sont aussi les parties du rapport.

La première tâche consiste à développer un nouveau service qui enrichira la fonctionnalité existante de synthèse vocale du robot. L'objectif est de programmer de petits mouvements de la tête et des bras du robot pendant qu'il parle, afin de renforcer l'expressivité et le réalisme de ses interactions.

La deuxième tâche se concentre sur les discussions entre le robot et un humain. Grâce à la fonction de reconnaissance de la parole, intégrée à QTrobot, il est possible de programmer des échanges simples basés sur des mots-clés. C'est ainsi que sera élaborée une application qui permettra à l'utilisateur d'interroger les bases de données de Wikipedia pour obtenir des informations pertinentes. Le robot récitera ensuite le résumé de l'article correspondant en utilisant la fonctionnalité de synthèse vocale précédemment développée.

Enfin, la troisième tâche implique l'utilisation de la carte audio ReSpeaker Mic Array v2.0 intégrée à QTrobot. Cette carte dispose de six microphones et d'un processeur DSP[4]. permettant de localiser la direction de la source sonore. Il est donc possible de développer une fonctionnalité pour que le robot tourne la tête dans la direction supposée du locuteur et d'évaluer la précision de ce dispositif.

Ce projet allie créativité, programmation et évaluation technique pour améliorer l'interaction entre QTrobot et les hommes, en rendant les échanges plus naturels et captivants.

I. Prise de parole vivante

A. Conception

QTrobot dispose de plusieurs services lui permettant de discuter avec son interlocuteur. Cependant, les conversations menées par QTrobot sont peu vivantes : en effet, le robot ne réalise aucun mouvement lorsqu'il parle. Le service Talk de ROS permet de lui ajouter des expressions faciales comme un sourire ou un clignement des yeux, mais ne permet pas de faire se mouvoir le robot.

C'est dans cette optique qu'intervient la conception de la fonction "Prise de parole vivante". Cette dernière a pour but de faire faire au robot des mouvements des bras et de la tête lorsqu'il prend la parole. Il est important d'assurer ce service dans le contexte d'utilisation du robot car ce dernier doit par exemple permettre d'habituer les enfants atteints de troubles autistiques à avoir des interactions sociales. Il faut donc un maximum de réalisme pour y parvenir.

Dans cette optique, le choix des moteurs à utiliser est important. La figure suivante (Figure 1) montre les différents moteurs disponibles et leur sens de fonctionnement.

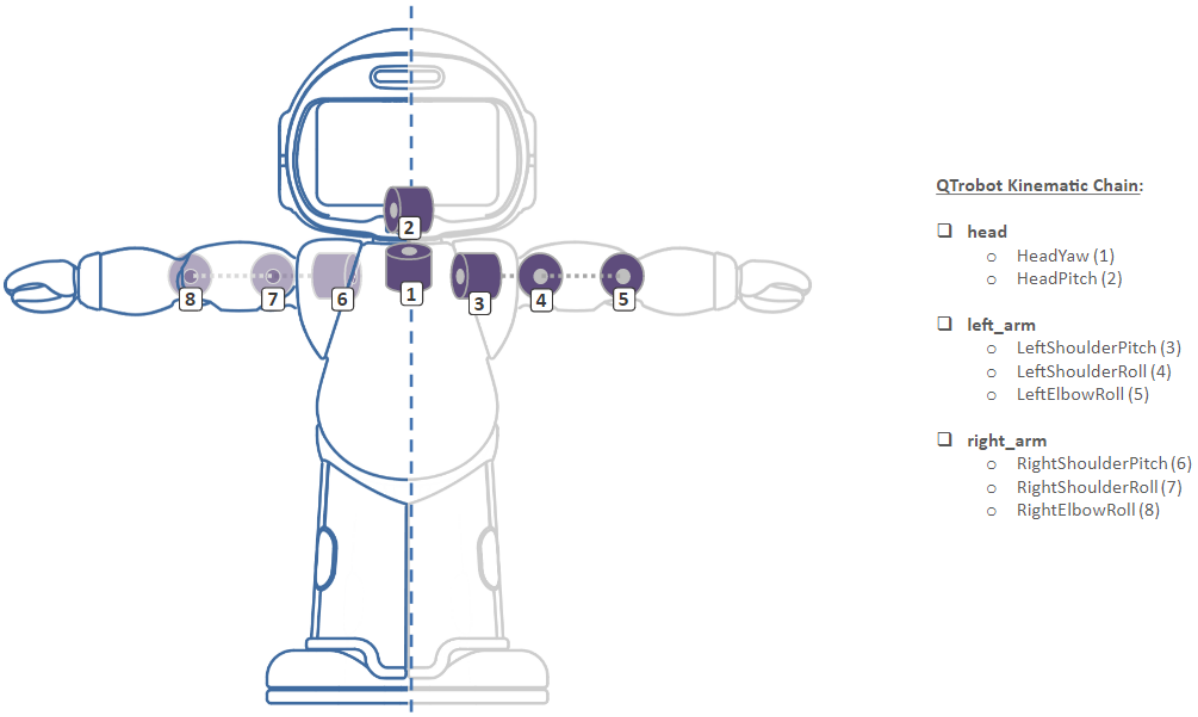


Figure 1 - Représentation des positions et sens des moteurs de QTrobot [5]

Pour ce qui est des mouvements des bras, la fonction utilise les moteurs ShoulderPitch et ShoulderRoll, et ce, pour les bras droit et gauche. En ce qui concerne les mouvements de la tête, elle utilise les deux moteurs disponibles, c'est-à-dire HeadYaw et HeadPitch.

La principale notion à prendre en compte avec les moteurs est leurs limitations d'angle. La figure ci-dessous montre les différentes limitations à prendre en compte.

Figure 2 - Limitations d'angle des moteurs [5]

La fonction utilise des angles faibles pour les mouvements de la tête afin de garantir un certain réalisme. Elle se restreint aux intervalles $[-75, -40]$ pour les moteurs ShoulderRoll et $[-75, 0]$ (respectivement $[75, 0]$ pour l'autre bras) pour les moteurs ShoulderPitch, tout cela ayant été testé pour délimiter les intervalles qui donnaient des mouvements naturels.

B. Réalisation

La fonction utilise trois publishers pour commander respectivement la tête, le bras droit et le bras gauche. Un publisher est un mécanisme ROS permettant d'envoyer des commandes à un composant robotique, sous forme de messages dans un canal de communication dédié. Dans ces derniers, le service publie des valeurs aléatoires d'angle pour chaque moteur. Les intervalles à l'intérieur desquels on choisit les valeurs aléatoirement sont donnés figure 3 :

```
head_yaw_ref = random.randrange(-15.0,15.0)
head_pitch_ref = random.randrange(-7.0,7.0)
RightShoulderPitch_ref = random.randrange(-75,0)
RightShoulderRoll_ref = random.randrange(-75,-40)
LeftShoulderPitch_ref = random.randrange(0,75)
LeftShoulderRoll_ref = random.randrange(-75,-40)
```

Figure 3 - Intervalles utilisés pour définir les positions à rejoindre

Le programme ajoute ensuite un temps de pause aléatoire compris entre 0,5 et 3 secondes pour rendre le tout plus naturel en créant des mouvements qui ne s'enchaînent pas toujours à la même fréquence. Enfin, il définit une certaine vitesse de rotation des moteurs que l'on peut modifier pour faire varier la manière avec laquelle le robot interagit avec les autres.

Lorsque l'utilisateur coupe le nœud ROS du programme, les moteurs de QTrobot rejoignent une dernière position qui correspond à l'état initial de ce dernier.

[rédacteur : Gabriel CHASTANET]
[relecteur : Victoria ANDRE, Maissa BEJI]

C. Validation

La validation est ici centrée sur des tests subjectifs. 13 personnes ont observé des situations filmées dans lesquelles QTrobot effectue le programme "prise de parole vivante". Le choix de filmer les interactions a été fait pour simplifier la mise en œuvre des tests car ce moyen ne nécessite pas que les personnes se rendent au labo. Ce choix peut toutefois légèrement fausser le sondage parce qu'il montre QTrobot dans un cadre parfait de fonctionnement.

Les 13 personnes choisies sont des étudiants d'IMT Atlantique. Ils ont donc environ 20 ans et sont en majorité des hommes (on compte deux femmes sur 13). Aucun élève international ne fait partie du groupe de testeurs. Le faible nombre de personnes interrogées ne permet pas de faire des hypothèses sur de potentielles corrélations entre les caractéristiques du groupe de testeurs et les résultats du sondage. Ces points pourraient cependant être étudiés plus en détails lors de futurs projets.

Les testeurs ont pu observer le robot réaliser des mouvements aléatoires et juger de leurs caractères naturel, fluide et cohérent, ainsi que de la pertinence des différents niveaux d'intensité. Les résultats sont proposés ci-dessous :

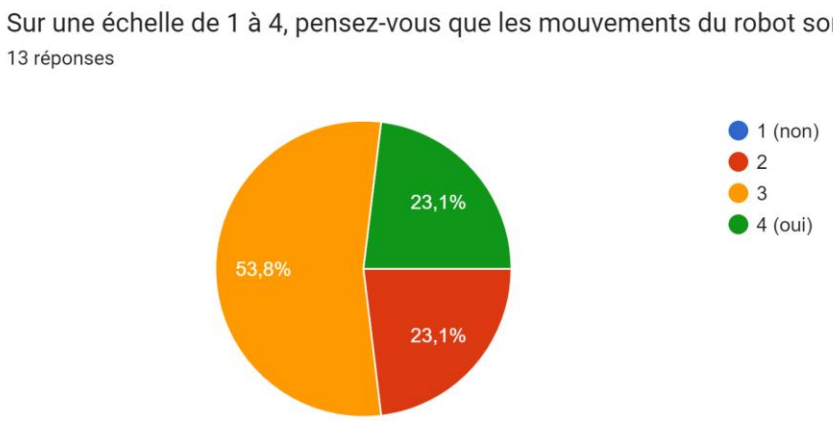


Figure 4 - Mouvements naturels

La figure 4 montre que la majorité (76,9%) des gens trouve que les mouvements de QTrobot sont naturels. Ce nombre est proche du résultat attendu dans le cahier des charges, mais il dépend également des valeurs obtenues pour la fluidité et la cohérence.

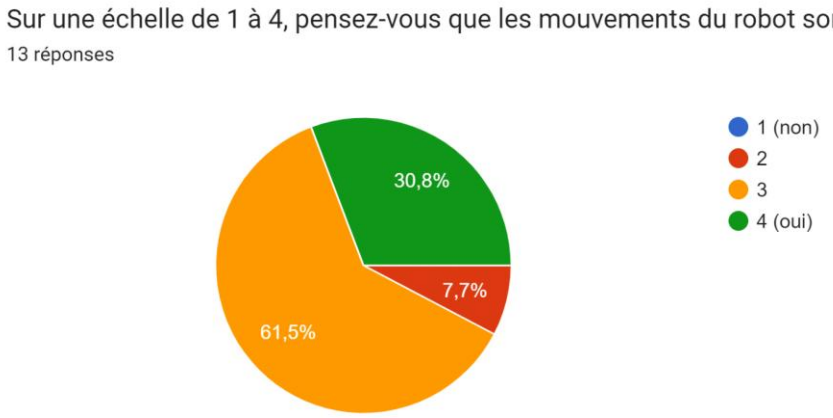


Figure 5 - Mouvements fluides

La figure 5 montre que la grande majorité (92,3%) des gens trouve que les mouvements de QTrobot sont fluides. Ainsi, il semble que le ressenti global lors d'un échange ne soit pas impacté par les légères saccades quasi inévitables du robot.

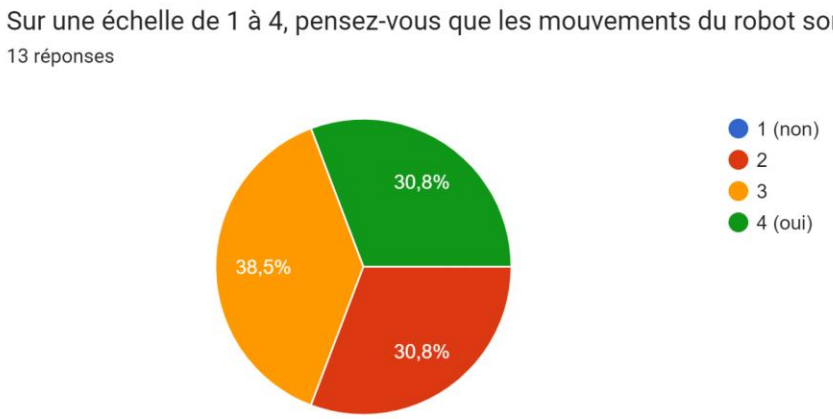


Figure 6 - Mouvements cohérents

La figure 6 montre une répartition beaucoup plus uniforme par rapport à la cohérence des mouvements du robot. Cet aspect semble logique pour un fonctionnement de mouvements aléatoires. Les positions futures des membres du robot ne sont pas choisies en fonction des mouvements précédents, ce qui entache parfois la cohérence globale du résultat.

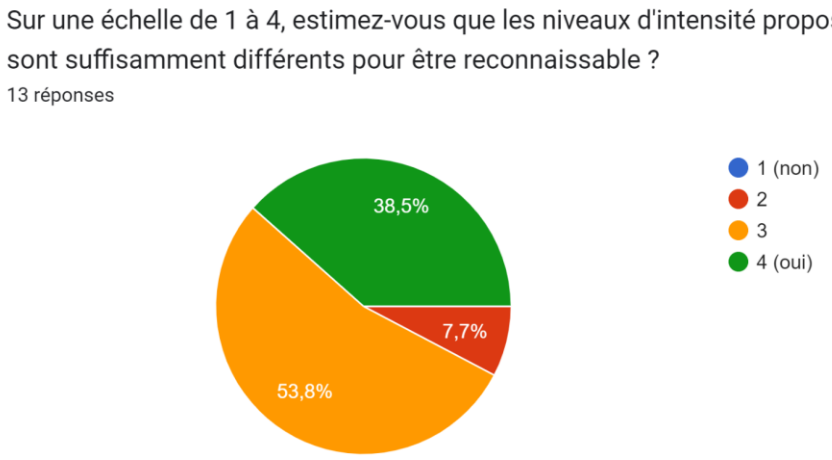


Figure 7 - Niveaux d'intensité

Enfin, la dernière figure décrit la perception des différents niveaux d'intensités disponibles et permet de s'assurer que, de par le résultat positif, ces niveaux ont un réel impact lors d'un échange avec QTrobot. Il y a en effet 92,5% des gens qui affirment reconnaître différents niveaux d'intensité, ce qui valide ce point vis-à-vis de la valeur attendue dans le cahier des charges.

En conclusion, de manière générale sur les trois premiers points, on compte en arrondissant une moyenne de 80% de bonnes réponses, ce qui valide le test vis-à-vis

du cahier des charges. De même pour le dernier graphique, nous pouvons valider le test car nous obtenons plus de 80% de réponses positives. Nous sommes donc satisfaits de cette fonction.

Maintenant que le robot effectue des gestes lorsqu'il parle et est donc moins statique, il est possible de mettre cette fonction en application en implémentant une phase de dialogue. Ainsi, le robot va pouvoir demander à son interlocuteur d'effectuer une recherche Wikipédia tout en faisant des gestes.

II. Effectuer une recherche Wikipédia

A. Conception

L'objectif de cette fonction est de permettre à l'interlocuteur de réaliser des recherches internet en soumettant des mots clés au robot, ce qui permettra aux utilisateurs d'enrichir leur connaissances d'une manière interactive.

Pour concevoir cette fonction, le robot doit être en mesure de reconnaître les mots clés prononcés par l'utilisateur et de les convertir en texte. Ceci est assuré par le microphone (figure 8) intégré à QTrobot.

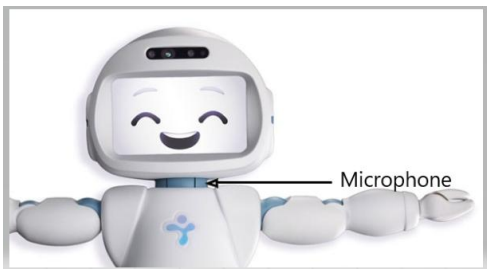


Figure 8 - Position du microphone sur QTrobot

Ensuite, le robot utilise une API pour interroger la base de données de Wikipédia, cette API permet d'interagir avec les serveurs de Wikipédia afin de récupérer la page qui correspond au mot-clé.

Finalement, QTRobot doit lire le résultat correspondant à la recherche souhaitée en utilisant une synthèse vocale qui convertit un texte en audio. La figure 9 montre où sont situés les hauts parleurs de QTrobot.

Figure 9 - Position des hauts parleurs de QTrobot

La figure 10 résume les étapes de la fonction “Effectuer une recherche Wikipédia”.

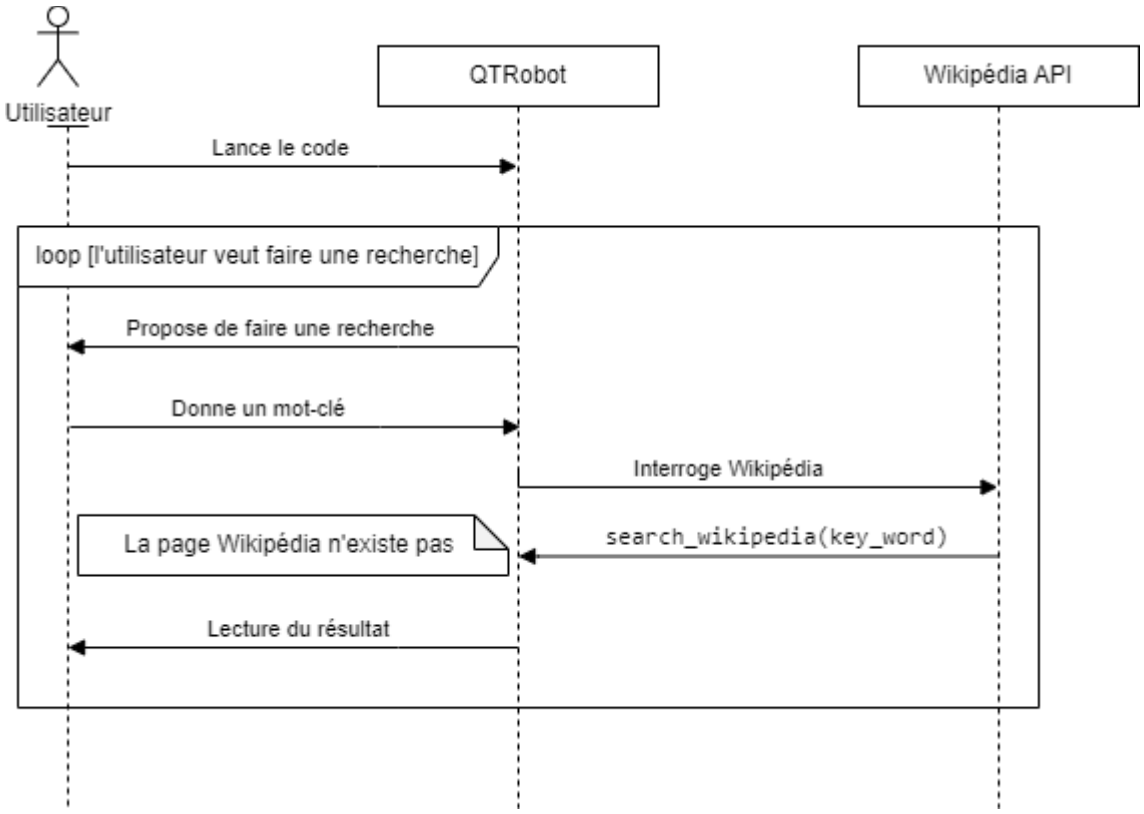


Figure 10 - Diagramme de séquences pour le cas d’utilisation “Effectuer une recherche Wikipédia”

B. Réalisation

La fonction est réalisée en intégrant deux modules :

- Module d’interaction avec l’utilisateur

Ce module utilise deux services :

- Service de synthèse vocale "**/qt_robot/speech/say**" du package "**qt_robot_interface.srv**" qui permet de convertir un texte en parole. Ce service est appelé par la fonction "**SpeechSay**". Il est principalement utilisé pour lire le résultat de recherche trouvé sur Wikipédia [6].

- Service de reconnaissance vocale "**/qt_robot/speech/recognize**" du package "**qt_vosk_app.srv**" [7], qui est appelé par la fonction "**recognize**" pour écouter les paroles de l'interlocuteur. Ensuite, le signal audio récupéré est transcrit, ce qui permet d'extraire le mot-clé prononcé par l'utilisateur.

- Module d'interaction avec Wikipédia

  Ce Module utilise la bibliothèque "**wikipediaapi**" [8] afin d'interroger l'API de Wikipédia. Il comprend une fonction "**search_wikipedia(key_word)**". Cette fonction est configurée pour effectuer la recherche en français. Elle commence par chercher la page Wikipédia adéquate. Si la page existe, plusieurs informations peuvent être extraites (titre, résumé …), la fonction récupère le résumé et sélectionne les 4 premières phrases afin de respecter la limite de temps (30s) imposée par le cahier des charges.

Le programme s'exécute en boucle tant que l'utilisateur souhaite effectuer des recherches.

C. Validation

Pour valider la fonction "Effectuer une recherche Wikipédia", 10 tests techniques ont été réalisés, en variant les mots clés :

- Vérification des réponses : pour tous les tests effectués, si la page Wikipédia existe, le résultat donné par QTrobot doit être conforme à la page Wikipédia correspondante.
- Temps de lecture du résultat : en imposant un nombre limité de phrases, le robot ne dépasse pas 30s. Pour les tests effectués, ce temps varie entre 20s et 29s.

- Temps de traitement du mot clé : Le temps mis par le robot avant de lire l'article sur Wikipédia dépasse 1s. La contrainte B.1 (Réagir rapidement) n'est donc pas validée. Ce temps provient principalement de la requête vers l'API Wikipédia et le traitement du texte.

Les tests ont démontré que la fonction est fiable, cependant elle n'est pas fluide (temps de traitement du mot clé). Ces observations conduisent à des pistes pour améliorer la performance de la fonction.

Par ailleurs, il est à souligner qu'il est important d'articuler correctement et de parler de manière claire pour maximiser la précision de la reconnaissance vocale. Il faut également attendre 2s avant de prononcer le mot-clé. Ce délai permet au robot de préparer le module de reconnaissance vocale.

D. Pistes d'amélioration

Pour une meilleure expérience d'interaction avec QTrobot, il semble judicieux d'améliorer les deux points suivants :

- Une modification de la fonction "**search_wikipedia**" de manière à optimiser le temps mis pour interroger Wikipédia.

- Accompagner la lecture et le temps de recherche des informations sur la page Wikipédia par des gestes appropriés et en harmonie avec les paroles du robot afin d'ajouter une dimension plus vivante à la fonction.

Le robot sait maintenant dialoguer avec quelqu'un (le terme dialoguer étant ici assez restrictif puisqu'il s'agit de l'échange par rapport à la recherche Wikipédia), mais il est possible d'encore améliorer son attitude lorsqu'il parle en le faisant suivre son interlocuteur du regard. C'est ainsi la dernière fonction qui sera implémentée.

III. Suivre l'interlocuteur du regard

A. Conception

Dans le but de rendre QTrobot plus interactif et humain, le robot doit suivre attentivement la personne qui parle en orientant sa tête dans la direction du son détecté, créant ainsi une interaction plus naturelle et immersive.

QTrobot est équipé d'une carte audio ReSpeaker Mic Array v2.0 qui comprend six microphones et un processeur DSP [4]. Cette carte permet la détection et

l'évaluation de la direction de la source sonore, et cette information est accessible via le service sound_direction en utilisant le framework ROS.

La figure 11 montre les interactions entre les différents modules de la fonction **‘sound_direction’.** En effet, lorsque l’utilisateur parle, le module “détection du son” envoie l’angle au module “contrôle des moteurs” qui pilote. Le HeadYaw est le moteur en charge de faire tourner la tête sur l’axe (Oz).

Finalement, le HeadYaw envoie sa position actuelle pour connaître le nouvel angle à renvoyer.

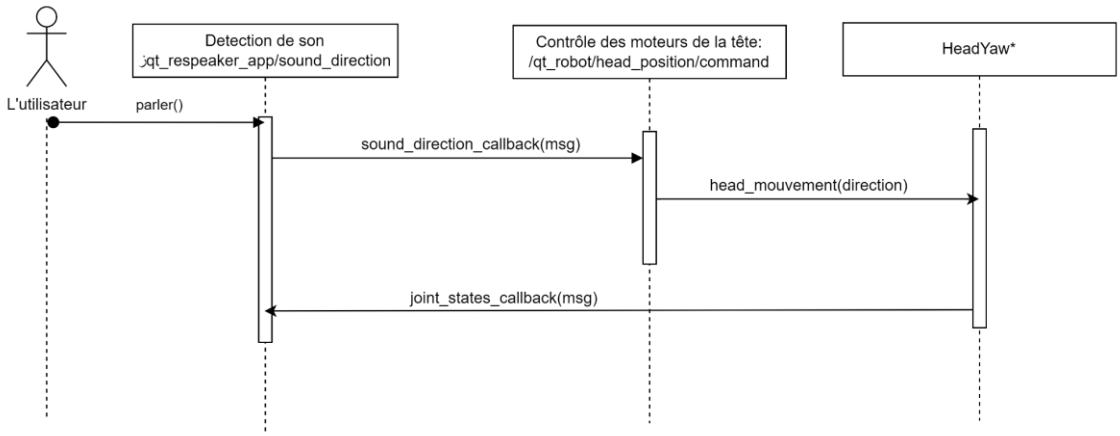


Figure 11 - Diagramme de séquence de la fonction sound_direction

B. Réalisation

La fonction consiste à implémenter en Python un nœud ROS qui s’appelle **‘direction_son’** pour détecter la direction de son, traiter les données et effectuer les mouvements de tête nécessaires.

Le nœud **"direction_son"** est conçu pour interagir avec différents composants du système robotique et réagir aux informations sur la direction du son détecté. Le nœud utilise plusieurs abonnements (**rospy.Subscriber**) à des sujets ROS pour recevoir les données pertinentes, tels que les messages de position des joints, la direction du son et l'indicateur de parole en cours.

Le nœud utilise principalement deux fonctions : "**joint_states_callback**" et "**sound_direction_callback**". La première consiste à envoyer les positions courantes des joints de HeadYaw. Quant à la deuxième, elle est appelée à chaque fois lorsqu’un

message qui contient la valeur d'angle de la direction de son est envoyé sur le sujet '**/qt_respeaker_app/sound_direction**'. Il faut effectuer quelques opérations sur la valeur prenant en considération la rotation du microphone avec la tête ainsi que les limitations d'angle de moteur **'headYaw'** ([-60°,60°]). La nouvelle position est publiée sur **'/qt_robot/head_position/command'** via **rospy.publisher** pour effectuer le mouvement correspondant.

Formule de calcul de l'angle :

Le microphone est placé sur le côté gauche de la tête du robot. le repère du microphone $R_2(O,\widehat{x_2},\widehat{y_2},\widehat{z_2})$ est en rotation de $\frac{\pi}{2}$ et d'axe $\widehat{z_2}$ par rapport au repère de la tête $R_1(O,\widehat{x_1},\widehat{y_1},\widehat{z_1})$ qui est lui-même en rotation d'angle α (la position actuelle de HeadYaw) et d'axe $\widehat{z_1}$ par rapport au repère de châssis $R_0(O,\widehat{x_0},\widehat{y_0},\widehat{z_0})$.

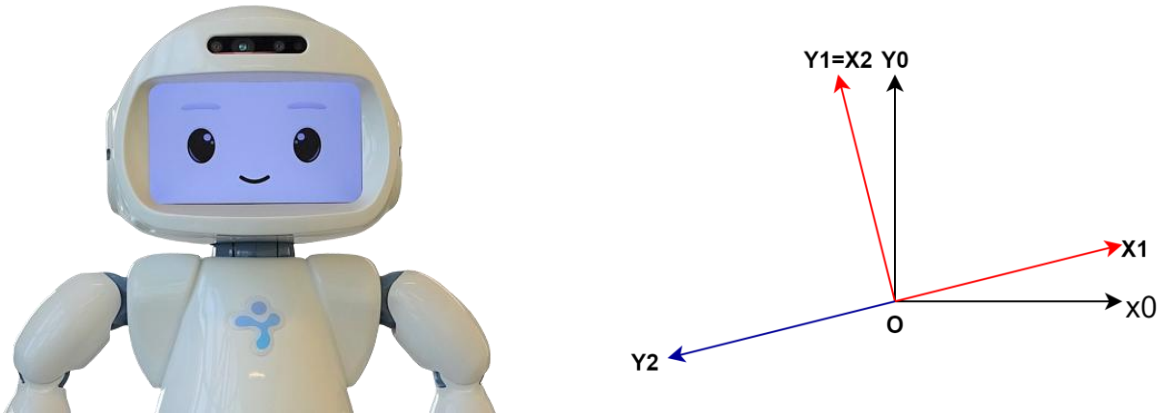


Figure 12 - Représentation des repères de robot sur le plan (x,y)

C. Validation

Pour évaluer la précision du module 'sound_direction', une expérience a été réalisée en effectuant 10 mesures sur un son émis dans une direction connue. L'objectif était de déterminer à quel point les mesures du module correspondaient à la direction réelle du son. Les tests sont faits dans le labo sur les angles respectifs 210°, 225°, 240°, 270°, 300°, 315°, 330° dans le repère du microphone.

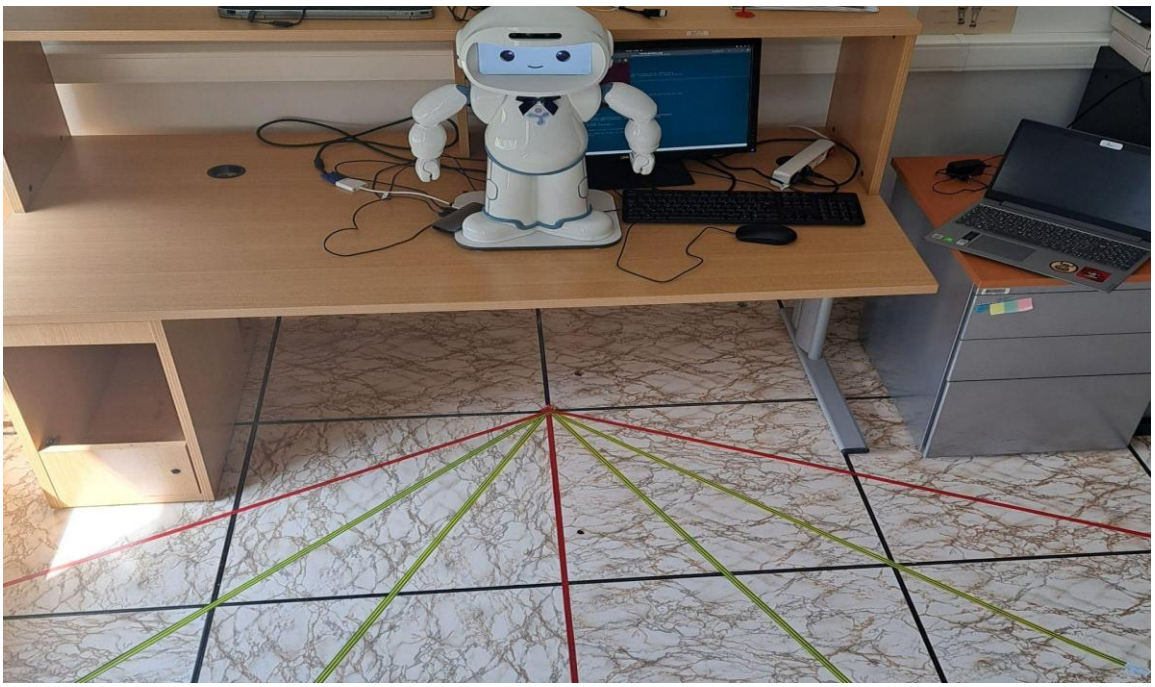


Figure 13 - Directions du son testées

Les résultats de l'expérience sont regroupés dans le graphe à barres suivant :

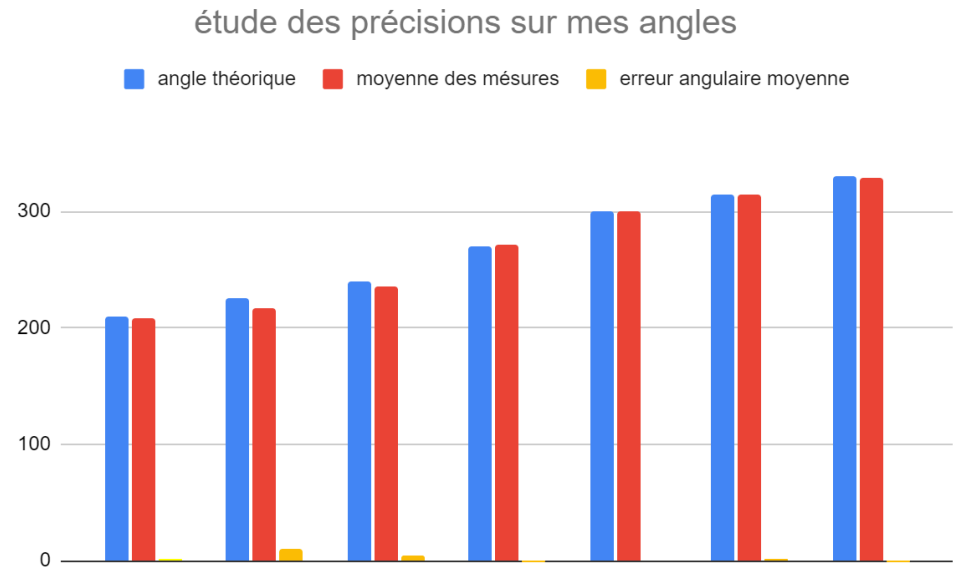


Figure 14 - Comparaison entre l'angle théorique et la moyennes des mesures ainsi que l'erreur angulaire moyenne

L'analyse des données montre que :

- Les angles mesurés par le robot présentent des variations par rapport aux angles théoriques. En effet, pour chaque angle théorique de l'ensemble (210°,

225°, 240°, 270°, 300°, 315°, 330°), les moyennes ont donné respectivement 208.9°, 216.4°, 235.6°, 271.5 °, 299.9°, 313.9° et 328.4°.

- L'erreur angulaire moyenne, étant la différence moyenne entre les angles théoriques et les angles mesurés, varie pour chaque angle entre -1.65° et 9.46°. La majorité des mesures ont une erreur angulaire moyenne en valeur absolue inférieure à 2°. Les mesures pour l'angle 225° ont l'erreur angulaire moyenne la plus grande (9.46°) et qui dépasse le seuil de $\pm 5°$.
- Le taux de réussite de la détection est calculé en utilisant un seuil de tolérance de 5°. Selon l'expérience, quatre angles présentent un taux de réussite 100%, deux angles qui ont un taux de réussite de 90% et un seul angle avec un de réussite très faible 10%.

Ainsi, le système de détection de direction du son du robot semble avoir une précision raisonnable, avec des erreurs angulaires moyennes généralement proches de zéro ou inférieures à 5 degrés. Cependant, il est important de noter que les résultats peuvent varier en fonction de plusieurs facteurs, notamment la configuration de l'environnement (présence de murs réfléchissants ou d'obstacles). De plus, la clarté et la qualité de la détection peuvent être influencées par la façon dont le son est émis.

Gestion de projet

A. Outils de restitution

Pour la gestion de notre projet, nous avons choisi de travailler avec GitHub* qui est un service web d'hébergement et de gestion collaborative de développement de logiciels, qui s'opère avec un programme de gestion de versions appelé Git. Ce dernier a permis de créer un dépôt qui consiste en un dossier contenant tous les fichiers du projet.

Ce dossier contient trois types de fichier. La licence afin de rendre le projet Open Source et de permettre à tous les membres du groupe de modifier le projet. Ensuite, il y a les fichiers correspondants aux codes des différentes fonctions que nous avons implémentées sur le robot. Et pour terminer, le fichier README.md qui permet d'afficher sous ces différents fichiers du texte avec des images expliquant le projet, l'organisation des différents fichiers avec leur utilité et enfin comment les exécuter.

B. Outils de communication interne

Pour faciliter la communication au sein de l'équipe, l'outil principal utilisé est “Messenger". C'est sur cette plateforme que l'équipe partage les informations nécessaires, telles que les horaires des réunions et l'avancement du projet.

Un espace de stockage partagé ‘Drive’ est utilisé pour organiser les documents et faciliter la rédaction du rapport technique, du cahier des charges ainsi que des feuilles de calcul pour les expériences.

Les échanges avec le tuteur se font principalement par e-mail et à travers les réunions dans le laboratoire.

Conclusion

À l’issue des trois phases de validation, il est possible de conclure que le projet a été une réussite. En effet, par rapport à la prise de parole vivante, le cahier des charges a été globalement respecté, les mouvements du robot sont fluides et cohérents d’après la plupart des utilisateurs. Ensuite, il est possible d’effectuer une recherche Wikipédia sur n’importe quel thème, le robot répondra à son interlocuteur. Pour terminer, le robot a été rendu plus naturel et vivant en tournant la tête vers son interlocuteur en fonction de sa position. De plus, le cahier des charges a bien été respecté par rapport aux écarts de précision de l’angle de l’orientation.

Ainsi, ce projet dans le domaine de l'interaction humain-robot, axé sur l'amélioration des fonctionnalités du robot humanoïde QTrobot, a réussi à rendre les interactions avec le robot plus engageantes et naturelles pour les humains.

Il serait possible d’améliorer encore les interactions entre ce robot et les hommes en ajoutant, de même que la fonction Wikipédia, d’autres dialogues comme des jeux mathématiques. Il serait aussi très intéressant de pouvoir donner des émotions au robot. En fonction de l’émotion qu’on donnerait au robot, il pourrait avoir des expressions faciales et des gestes adaptés. Cela ouvrirait sur un tout autre aspect de la robotique, tout aussi intéressant.

Références bibliographiques

[1] LuxAI. QTrobot Documentation. 2022.
URL : https://docs.luxai.com/
(visité le 10/03/2023).

[2] ROS Wiki. LuxAI QTrobot. Fév. 2020.
URL : http://wiki.ros.org/Robots/qtrobot
(visité le 10/03/2023).

[3] ROS Wiki. Getting Started/Learning ROS.
URL : http://wiki.ros.org/ROS/StartGuide
(visité le 17/03/2023).

[4] LuxAI. QTrobot Audio processing and Microphone
URL : https://docs.luxai.com/docs/modules/microphone
(visité le 20/05/2023)

[5] LuxAI. QTrobot Motion and Actuators. 2022.
URL : https://docs.luxai.com/docs/modules/motors
(visité le 21/05/2023)

[6]LuxAI. QTrobot interfaces using ROS Services.
URL : https://docs.luxai.com/docs/tutorials/python/python_ros_services
(visité le 20/05/2023)

[7] LuxAI. QTrobot Offline speech recognition.
URL : https://docs.luxai.com/docs/tutorials/python/python_ros_vosk
(visité le 20/05/2023)

[8] Projet Wikipédia API.
URL : https://pypi.org/project/Wikipedia-API/
(visité le 20/05/2023)

Glossaire

Git : logiciel local qui permet aux développeurs de sauvegarder instantanément leurs projets au fil du temps

GitHub : plateforme web qui intègre les fonctionnalités de contrôle de version de Git afin de pouvoir les utiliser en collaboration

Processeur DSP : (Digital Signal Processor) est un type spécifique de processeur conçu pour effectuer des opérations de traitement du signal numérique de manière efficace.

Publisher : est un composant logiciel qui envoie des messages à un topic spécifique.

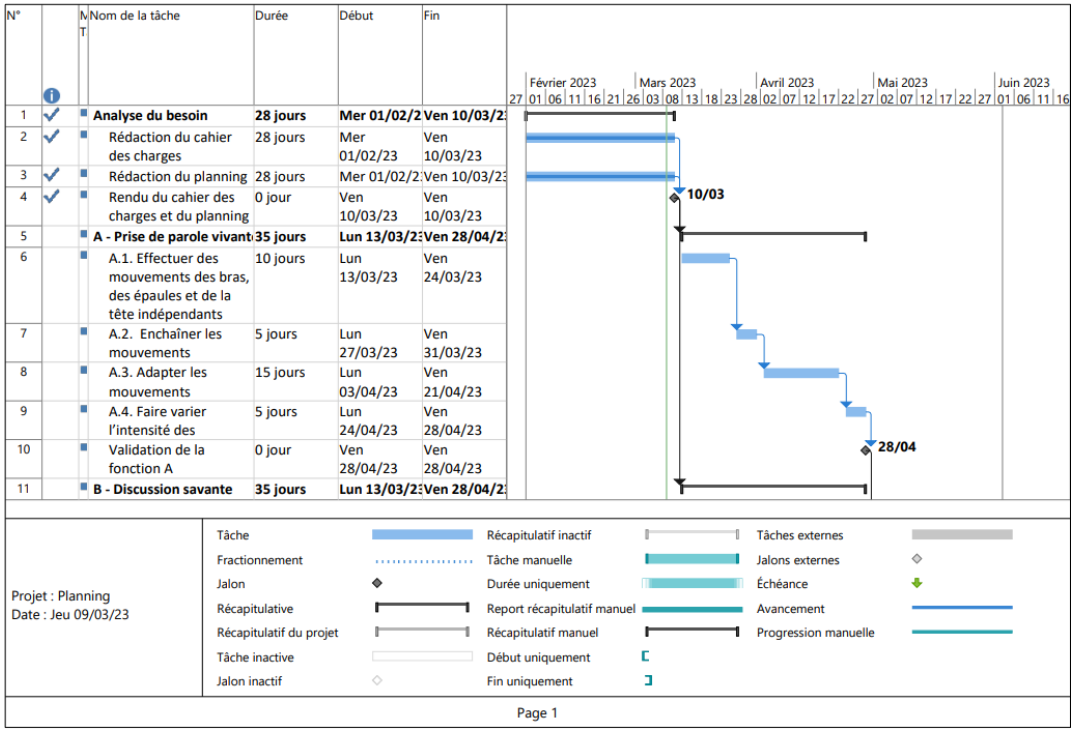
ReSpeaker Mic Array v2.0 : est un microphone à matrice conçu spécialement pour les applications d'IA vocale et de reconnaissance vocale.

ROS : (Robot Operating System) ensemble d'outils informatiques sous forme de logiciels libres open source, permettant de développer des logiciels pour la robotique.

Annexe

Planning prévisionnel et planning réel

Planning prévisionnel

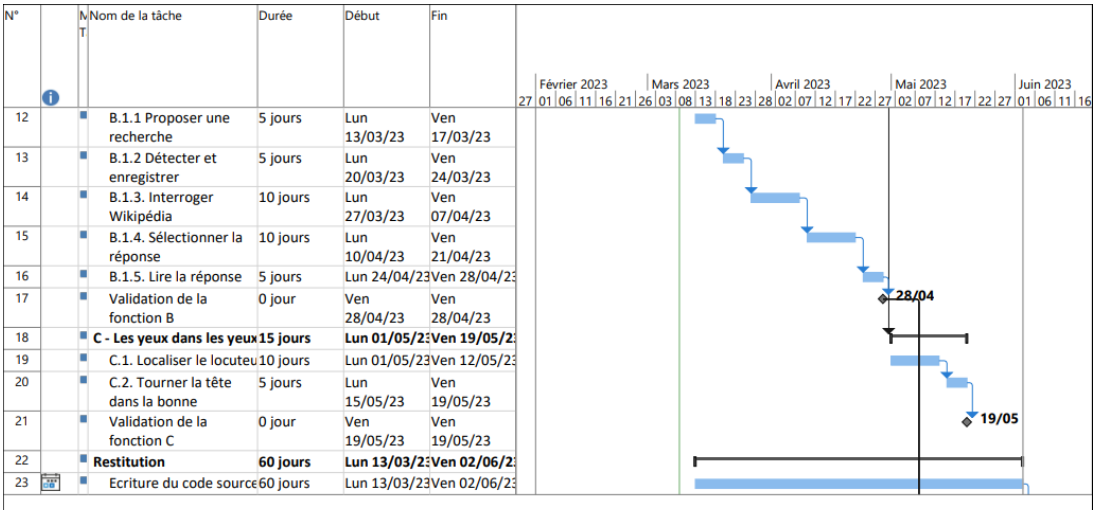


N°		M T	Nom de la tâche	Durée	Début	Fin
1	✓		**Analyse du besoin**	**28 jours**	**Mer 01/02/2**	**Ven 10/03/2**
2	✓		Rédaction du cahier des charges	28 jours	Mer 01/02/23	Ven 10/03/23
3	✓		Rédaction du planning	28 jours	Mer 01/02/2	Ven 10/03/23
4	✓		Rendu du cahier des charges et du planning	0 jour	Ven 10/03/23	Ven 10/03/23
5			**A - Prise de parole vivant**	**35 jours**	**Lun 13/03/2**	**Ven 28/04/2**
6			A.1. Effectuer des mouvements des bras, des épaules et de la tête indépendants	10 jours	Lun 13/03/23	Ven 24/03/23
7			A.2. Enchaîner les mouvements	5 jours	Lun 27/03/23	Ven 31/03/23
8			A.3. Adapter les mouvements	15 jours	Lun 03/04/23	Ven 21/04/23
9			A.4. Faire varier l'intensité des	5 jours	Lun 24/04/23	Ven 28/04/23
10			Validation de la fonction A	0 jour	Ven 28/04/23	Ven 28/04/23
11			**B - Discussion savante**	**35 jours**	**Lun 13/03/2**	**Ven 28/04/2**

Projet : Planning
Date : Jeu 09/03/23

Tâche	Récapitulatif inactif	Tâches externes	
Fractionnement	Tâche manuelle	Jalons externes	
Jalon	Durée uniquement	Échéance	
Récapitulative	Report récapitulatif manuel	Avancement	
Récapitulatif du projet	Récapitulatif manuel	Progression manuelle	
Tâche inactive	Début uniquement		
Jalon inactif	Fin uniquement		

Page 1

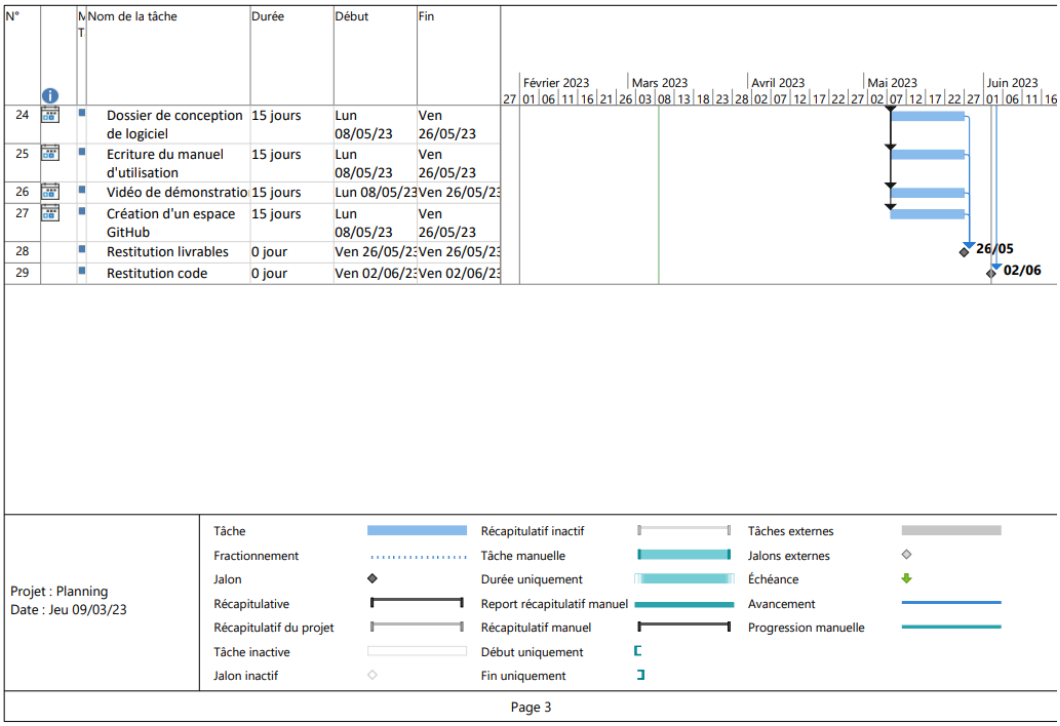


N°		M T	Nom de la tâche	Durée	Début	Fin
12			B.1.1 Proposer une recherche	5 jours	Lun 13/03/23	Ven 17/03/23
13			B.1.2 Détecter et enregistrer	5 jours	Lun 20/03/23	Ven 24/03/23
14			B.1.3. Interroger Wikipédia	10 jours	Lun 27/03/23	Ven 07/04/23
15			B.1.4. Sélectionner la réponse	10 jours	Lun 10/04/23	Ven 21/04/23
16			B.1.5. Lire la réponse	5 jours	Lun 24/04/23	Ven 28/04/23
17			Validation de la fonction B	0 jour	Ven 28/04/23	Ven 28/04/23
18			**C - Les yeux dans les yeux**	**15 jours**	**Lun 01/05/2**	**Ven 19/05/2**
19			C.1. Localiser le locuteu	10 jours	Lun 01/05/23	Ven 12/05/23
20			C.2. Tourner la tête dans la bonne	5 jours	Lun 15/05/23	Ven 19/05/23
21			Validation de la fonction C	0 jour	Ven 19/05/23	Ven 19/05/23
22			**Restitution**	**60 jours**	**Lun 13/03/2**	**Ven 02/06/2**
23			Ecriture du code source	60 jours	Lun 13/03/23	Ven 02/06/23

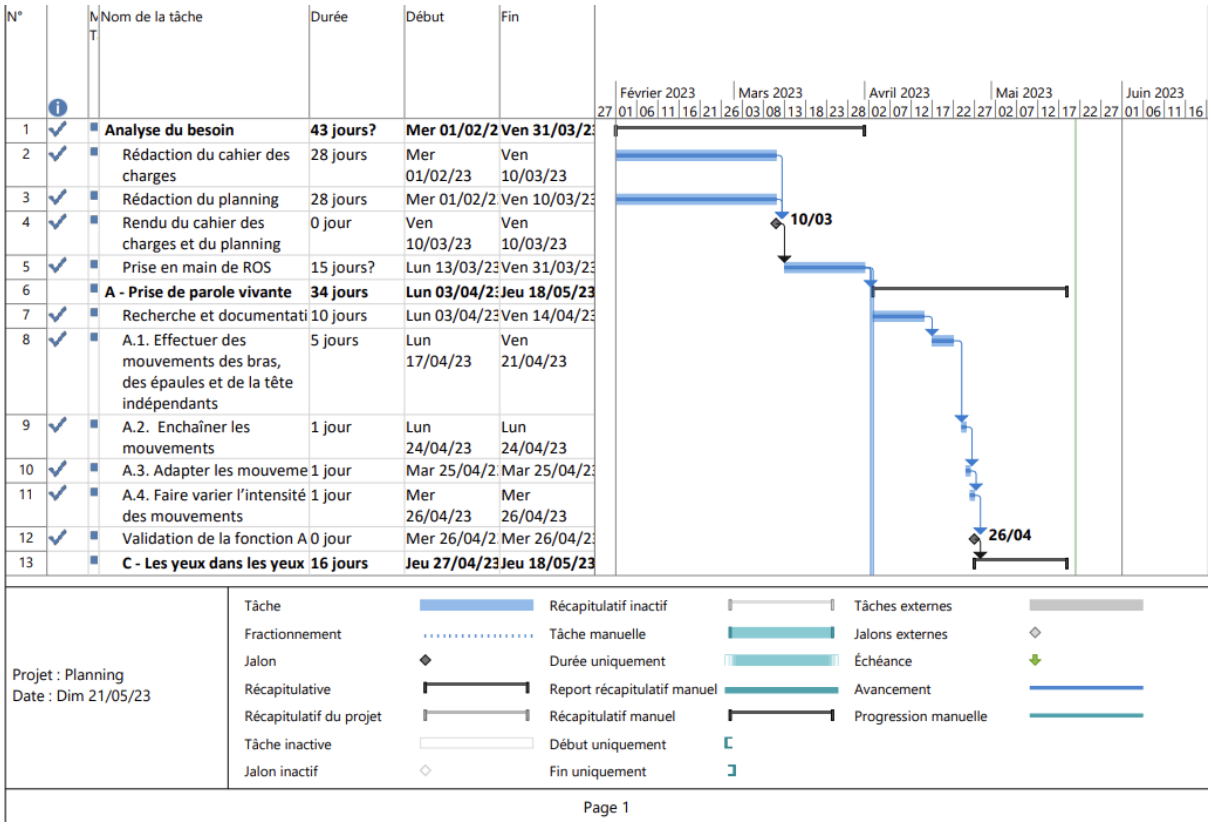
Projet : Planning
Date : Jeu 09/03/23

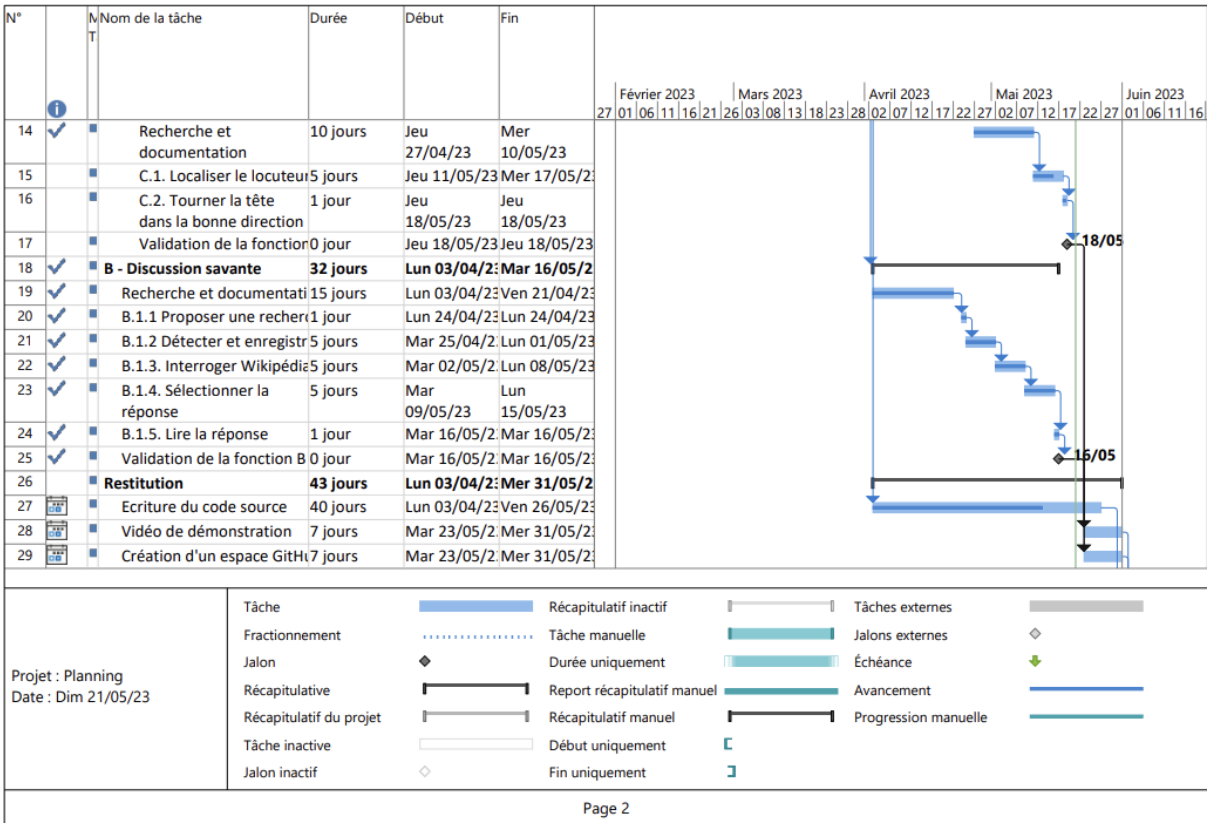
Tâche	Récapitulatif inactif	Tâches externes	
Fractionnement	Tâche manuelle	Jalons externes	
Jalon	Durée uniquement	Échéance	
Récapitulative	Report récapitulatif manuel	Avancement	
Récapitulatif du projet	Récapitulatif manuel	Progression manuelle	
Tâche inactive	Début uniquement		
Jalon inactif	Fin uniquement		

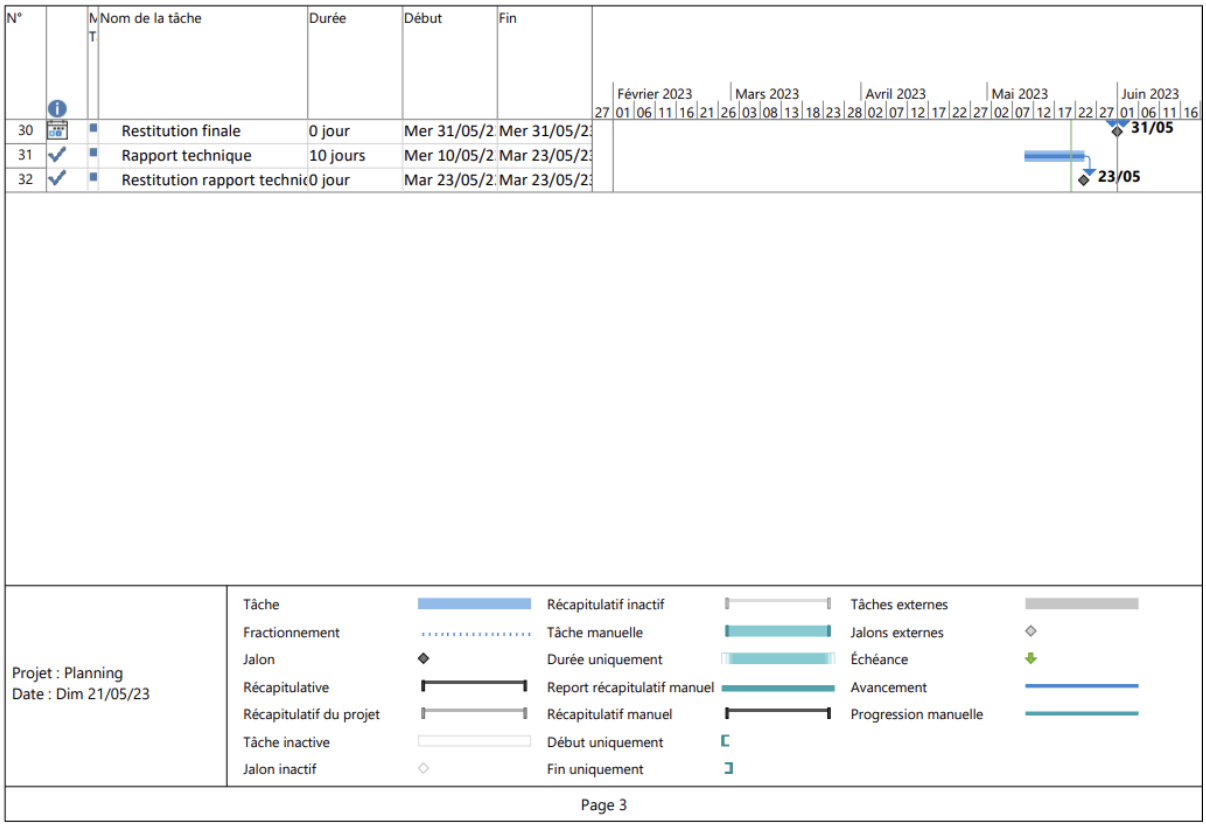
Page 2



Planning réel







Les deux plannings en annexe sont : le planning prévisionnel du projet et le planning de ce qui a été réalisé en réalité. Les plannings prévisionnel et réel sont sensiblement les mêmes. En effet, la structure en trois axes a été conservée. L'équipe

chargée de la fonction A a comme prévu terminé son travail en avance en comparaison de la fonction B, c'est pour cela qu'elle a enchaîné sur la fonction C qui reprend des notions communes à la fonction A.

La différence réelle entre les deux plannings est le fait que beaucoup de temps a été dépensé dans la documentation, à la fois sur le fonctionnement de ROS en général et sur la compréhension de service précis à utiliser dans le code. La partie écriture des fonctions a donc été plus rapide que prévu dans la majorité des cas, mais il a fallu prendre en compte cette documentation dans le planning réel.

Enfin, les dates de livraison des différents livrables ont été ajustées avec tous les détails obtenus récemment, de même pour l'organisation adoptée pour cette fin de projet.