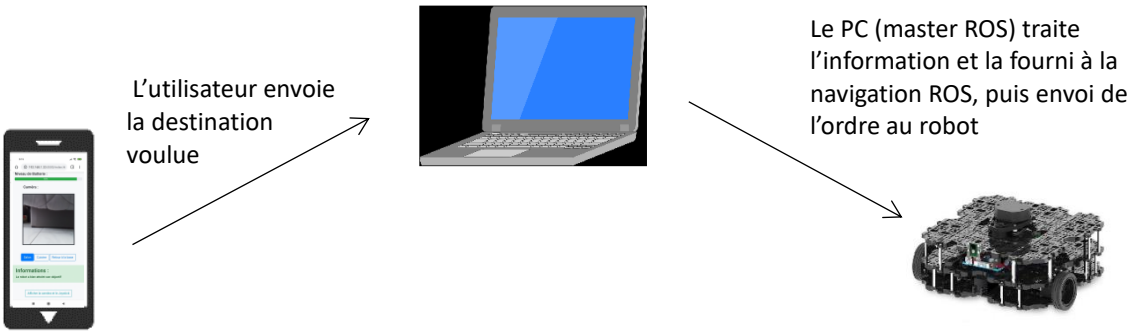
25/08/2020
ROLLAND
Killian

Perspectives et solutions

Ce document a pour but de donner des perspectives d'amélioration du travail que j'ai pu effectuer autour du TurtleBot et de l'autonomie du robot dans un contexte d'aide à la personne.

Une première problématique bien apparente est que pour utiliser le TurtleBot, il nous faut un PC. En effet, la Raspberry Pi ne peut pas contenir le Master ROS et que donc lorsque l'on pilote le robot avec les algorithmes que j'ai écrit, on se trouve dans la configuration suivante :



Ce problème vient du fait que c'est la configuration qui a été voulue par les constructeurs du robot (Robotis). J'ai moi-même essayé de migrer le Master ROS sur la Raspberry Pi, le problème est que la version de Linux présente sur la carte (Raspbian) ne peut pas contenir les programmes de navigation puisque certains packages (notamment *Intercative Markers*) ne peuvent pas être installés.

Plusieurs solutions pourraient être envisageables. On pourrait par exemple essayer de changer d'OS sur la Raspberry Pi pour en avoir un qui permet d'avoir tous les packages nécessaires au bon fonctionnement du système ROS. J'ai essayé avec une copie de Ubuntu 16.04 adapté pour Raspberry Pi : https://downloads.ubiquityrobotics.com/pi.html, cependant cela n'a pas porté ses fruits. Une autre idée de solution serait de changer la Raspberry Pi par une autre carte. On pourrait par exemple prendre une Odroid XU4 comme celle présente sur le meuble mobile qui pourrait prendre en charge un OS plus compatible pour un Master ROS et permet aussi de gagner en puissance de traitement de données. En effet, avec la Raspberry, on a la sensation que celle-ci manque un peu de puissance dans certains cas : par exemple, lors d'un déplacement en navigation, on remarque que le flux de la caméra devient très vite saturé et on observe une multitude « d'arrêts sur image ».

Le projet peut aussi être amélioré au niveau de la complexité de mise en place du système. En référence au fichier sur le Google Drive du projet :

« *Documentation/Document_deploiement_TurtleBot3.pdf* », on remarque facilement que cette mise en place est assez longue puisqu'il faut effectuer la cartographie, enregistrer les points voulus par le code, voire même devoir modifier l'interface graphique afin d'ajouter/enlever/modifier des boutons. Tous ces éléments pourraient être gérés par un algorithme qui procéderaient aux installations, puis demanderaient à l'utilisateur quelles actions il veut effectuer. Pour la partie « cartographie » de l'environnement, il existe des packages ROS qui permettent de faire explorer le robot tout seul l'environnement tout en

créant la carte. Cela permettrait d'éviter à l'utilisateur de devoir faire explorer le robot chaque recoin en utilisant une manette/un téléop. Cela permettrait aussi de s'assurer d'avoir une carte suffisamment apte à la navigation ROS du robot (exemple d'algorithme d'exploration : http://wiki.ros.org/frontier_exploration).

Cet algorithme pourrait aussi automatiquement remplacer les positions enregistrées du robot (voir le switch/case de l'algorithme dans le lien : *Chaire_M@D_Stage2020-Meuble_Mobile&TurtleBot_Killian_ROLLAND/catkin_ws/src/simple_navigation_goals/src/simple_navigation_goals.cpp* entre les lignes 115 et 195 et ligne 85 dans *battery_monitoring.cpp*).

Afin de donner une meilleure interaction entre le robot et l'utilisateur, la synthèse vocale est une bonne piste d'améliorations. J'ai pu expérimenter légèrement cette idée comme je le décris dans mon *Compte-Rendu Hebdomadaire n°9.*

En effet, cette méthode permettrait de mettre au courant l'utilisateur de l'état actuel du robot en lui énonçant des informations du type « Je me déplace jusqu'au lit », « je suis arrivé à destination », « je vais me recharger, je n'ai plus de batterie ».