Rapport IA Eve

Huber Robin

Kahlaoui-Guillaume Rayan

Loussert Mathis

Poncy-Bernard Oscar

Table des matières :

1) Développement :

Au début, nous avons divisé et réparti les principales caractéristiques du robot, les déplacements, les pinces et les différents capteurs. Nous avons commencé par des tests sur ces différentes parties pour comprendre leur fonctionnement et commencer par des méthodes simples.

Par la suite, nous sommes rentrés dans des méthodes plus complexes comme la détection de palet avec l'ultrason ou les déplacements asynchrones qui permettaient d'utiliser plusieurs méthodes en même temps.

Au fil du développement nous avons opté pour *RotationRapide* qui demande au deux moteurs des roues de tourner l'une en sens inverse de l'autre de façon synchronisée. Suite à de nombreux tests, nous avons décidé d'une constante pour la rotation des roues ce qui ne donne pas une rotation d'un angle exact mais qui limite l'impact dû aux phases d'accélération et de décélération qui faussent l'angle.

En parallèle du développement des méthodes, le programme principal 'main' ainsi que les autres parties importantes comme l'arrêt d'urgence du robot, le timer ou encore les constantes sont rédigés, mis au propre et liés toutes les méthodes entre elles.

Pour finir, nous avons décidé de dédier nos ressources sur la résolution des problèmes liée au programme. Par conséquent nous ne nous sommes pas attardé sur un programme qui permet le recalibrage de la boussole lors d'un tour complet.

2) Ce qui a marché :

Les tests et les premières écritures de méthode étaient convaincants, surtout quand les méthodes étaient créées toute seule, hors du main.

Les méthodes *avancerJusqua* et *moveTo* qui permettent au robot de se déplacer respectivement jusqu'à l'enclenchement du capteur pression et d'une certaine longueur fonctionnent correctement. Tout comme le capteur de pression et le capteur d'ultrasons.

La classe bouton d'arrêt d'urgence (*BAU*) marche parfaitement ainsi que le timer de 2 minutes et 30 secondes qui arrête le programme.

Le système de changement d'état marche très bien

Le thread d'ouverture des Pince fonctionne mais le programme peut sûrement être simplifié afin d'économiser des lignes.

3) Amélioration :

detection180, lors de nos tests, le robot détecte sous plusieurs angles au-dessus, soit au-dessous. Nous aurions pu ralentir le robot lors de la rotation pour être plus précis. Même si on prenait le plus de données possible par angle.

Le *detectPale*t aurait pu être amélioré avec un peu plus de test est de rectification, l'intervalle durant lequel il regarde est possiblement un peu trop grand. Ce qui baissé la precision et la discrimination palet/robot/mur. On aurait pu améliorer la discrimination en ajoutant une vérification pour confirmer que l'objet detecté n'est pas un robot adverse. Par exemple : une fois l'angle trouvé, se tourner a cet angle, vérifier que la distance n'a pas bougé et ainsi garder en mémoire d'une façon quelconque un second angle si jamais la première distance à changer.

Pour le allerChercherPalet(), au lieu d'ouvrir les pinces suffisamment proche du palet, le robot les ouvre quand il se dirige vers le palet quelque soit la distance. Les pinces sont donc trop souvent ouvertes et imposer une distance minimale d'ouverture aurait été un meilleur choix.

Nous aurions pu chercher un moyen d'avoir un angle exact à chaque utilisation de *rotationRapide*.

4) Problème rencontré et non résolu :

Lors de chaque test de déplacement ou test des capteurs, tout a fonctionné comme prévu. Mais lorsqu'on a tout regroupé dans les classes *brain* et *activators* de nombreux programmes ne marchaient plus comme on le voulait. Malgré de nombreuses tentatives de fixage.

Les tests sur le capteur couleur n'étaient pas assez précis et le fait qu'il ne détecte pas toutes les couleurs,ou alors de façon précise, nous a imposé de ne presque pas pouvoir l'utiliser.

La méthode *rotationRapide* nous a beaucoup freiné puisqu'à chaque lancement du programme le robot tourne aléatoirement sans paterne régulier permettant une rectification du code. Le programme qui utilise plusieurs threads devait ralentir le lancement de certaines parties et la rotation était différente en fonction de comment le reste du programme fonctionnait, comme la capture de valeur avec l'ultrason pendant la rotation.

La méthode *esquive* était difficile à rectifier sachant les problèmes liés à la rotation. Par conséquent, on ne peut pas critiquer correctement ces deux méthodes.