20/11/2023

Plan de tests de Pablo

BONNARDEL Adrien 12106324
DUEZ Valentine 12107343
ROMANET Clément 12213747
ROUME Emma 12125449

1. Introduction

L'objectif du document est de présenter les tests effectués pour garantir le bon fonctionnement de notre programme et son efficacité. Ainsi les méthodes de bas niveaux pourront être utilisées pour faire des méthodes plus complexes.

1.1. Objectifs et méthodes

Présenter le développement du logiciel dans son ensemble.

1.2. Documents de référence

Voici les documents ayant servi à la création de ce plan de test.

- Diagramme des classes
- Plan de développement
- Cahier des charges
- Exemple de plan de test fournit pas M.Pellier

1.3. Liste des tests

- UultrasonicSensor()
- ColorSensor()
- TouchSensor()
- Se déplacer (avancer et reculer)
- tourner()
- setVitesse()
- ouvrir/fermerPince()
- démarrage()
- recherchePalet()
- prendrePalet()
- ramenerPalet()

2. Tests unitaire

Pour chaque test, nous indiquerons à quel niveau de fonctionnement la méthode par le biais d'un code couleur.
🟩 La méthode fonctionne, n'a pas de contraintes (ou des contraintes inhérentes)
🟨 La méthode fonctionne, elle présente des contraintes
🟧 La méthode ne fonctionne partiellement
🟥 La méthode ne fonctionne pas

2.1. Sensors :

2.1.1. Description

Les tests unitaires de nos sensors nous permettent de savoir si les capteurs marchent correctement indépendamment les uns des autres. L'environnement de test est le terrain du match.
Nos 3 tests :

- UltraSonicSensor() : le test consiste à positionner le robot de manière statique à différentes distances.Il va ensuite envoyer des ondes qui vont se réfléchir sur un objet. Le temps que l'onde revienne au robot, cela définit la distance entre le robot et l'objet.

- TouchSensor() : mettre le robot en face d'un palet le faire avancer jusqu'à qu'il entre dans le palet et quand il y a collision le robot s'arrête grâce à la méthode. isPressedContinue() qui utilise isPressed.

- ColorSensor() : Mettre le robot sur les différentes lignes de couleurs et voir s' il reconnaît les différentes couleurs de chaque ligne.

2.1.2. Contraintes

Pour le test du ColorSensor() il faut l'effectuer le jour du match car cela dépend de la lumière ambiante de la salle du match.

2.1.3. Dépendances

Nous avons besoin d'effectuer le test de la classe MoteurRoues() pour pouvoir tester le TouchSensor(). Pour le TouchSensor(), nous avons besoin que le robot avance en ligne droite.

2.1.4. Procédure de test

Pour le test UltraSonicSensor(), on met un palet en face du capteur à une certaine distance et on regarde la valeur renvoyée si elle correspond bien à nos mesures.

2.2. MoteurPinces() :

2.2.1. Description

Pour la classe MoteurPinces(), on possède 2 méthodes :

- 🟩 ouverturePince()
- 🟩fermeturePince()

Pour ces deux méthodes nous avons fait des tests pour réussir à trouver le bon angle d'ouverture pour que le robot puisse prendre le palet et la bonne fermeture pour qu'il attrape le palet et le garde dans ses pinces le temps de l'amener dans le camp adverse.

2.2.2. Contraintes

Il faut prendre en compte la capacité d'ouverture et de fermeture des pinces, il ne faut pas les dérégler en les ouvrant trop ou en les fermant trop car sinon les pinces se superposent et se dérèglent, elles seront donc inutilisables jusqu'à la fin du tournoi.

2.2.3. Dépendances

Il n'y a aucune dépendance.

2.2.4. Procédure de test

Le robot doit être capable d'ouvrir ses pinces avec la valeur entrée en paramètre, cette valeur a été choisie de sorte que l'angle soit assez grand pour pouvoir mettre le palet entre les pinces. Il doit aussi être capable de fermer ses pinces avec la valeur entrée en paramètre, cette valeur a été choisie de sorte que les pinces attrapent le palet sans forcer pour ne pas dérégler les pinces.

2.3. MoteurRoues() :

2.3.1. Description

🟩 avancer()
Pour la méthode avancer() il suffit de vérifier si le robot avance de la distance entrée en paramètre.
🟩 reculer()
De même pour reculer on vérifie s'il recule de la distance qu'on lui a donnée
🟩 tourner()
Cette méthode permet au robot de tourner d'un angle donné, il faut vérifier si cela coïncide avec les mesures que nous avons entrées

🟩 setVitesse()
Pour cette méthode nous avons nous même quantifié ce que voulait dire vitesse rapide, moyenne lente et la meilleure, nous avons donc choisi les vitesses pour toutes ces catégories.

Pour toutes ces méthodes des tests ont dû être effectués avant car il a fallu faire coïncider le chiffre passé en paramètre avec la réalité, le robot ne parcourt pas la même distance, il fallait donc mesurer plusieurs fois pour avoir plusieurs mesures , puis ensuite faire des produits en croix pour pouvoir trouver la valeur à passer en paramètre (cf tableau de mesure). Toutefois malgré les nombreux tests réalisés nous avons toujours une légère marge d'erreur, nous avons donc pris la valeur qui était la plus souvent exacte.
De même pour les valeurs de la méthode tourner().

2.3.2. Contraintes

La distance donnée n'est pas la même que celle parcourue il a fallu que l'on mesure plusieurs fois puis ensuite nous avons fait des produits en croix pour avoir toutes nos distances, de même pour les angles de la méthode tourner().

2.3.3. Dépendances

Aucun test n'a à être mené avant

2.3.4. Procédure de test

Le robot est placé sur le terrain, on lui donne une distance à parcourir, s'il parcourt cette distance et s'arrête au bout de la distance parcourue, le test est validé. De même pour la méthode reculer().
De même pour la méthode tourner(), on entre un valeur pour trouver lorsqu'il tourne de 360°.

3. Tests d'intégration

3.1. Pour chaque test d'intégration :

3.1.1. Description

🟩 départ() consiste à récupérer le plus rapidement le premier palet, le robot avance tout droit, jusqu'à que le sensor touche soit true, tourne avance entre deux lignes puis retourne vers les buts, l'objectif de cette partie est de ne pas récupérer deux palets à la fois ce qui est interdit. Puis il avance tout droit vers le but, lorsqu'il passe la ligne blanche il relâche le palet.
🟩 recherchePalet(), le robot tourne sur lui-même en créant un tableau de valeur avec la distance des choses en face de lui. Un trop grand écart entre la distance précédente et la distance actuelle est considéré comme étant un palet.

🟨 prendrePalet(), après avoir repéré un palet, le robot avance vers le palet en vérifiant que la distance diminue, si ce n'est pas le cas il retourne sur recherchePalet(), sinon il avance jusqu'à son sensorTouch() soit true et ferme ses pinces.
🟨 ramenerPalet(), La méthode ramenerPalet() contient un compteur modulo 342 (360° dans la réalité). Au début de la partie, le compteur est initialisé à 0 degré, on sait qu'il est en face des cages donc tous les modulo 342, il sera en face des cages. Quand il récupère un palet, il va compenser l'angle restant pour retourner à un multiple de 342, donc on saura qu'il sera pile en face des cages. Une fois le palet récupéré, il avancera tout droit jusqu'à dépasser la ligne blanche puis il relâche le palet et se retourne de 180° réel.

3.1.2. Contraintes

Notre robot doit pouvoir tenir en batterie durant tout le tournoi.

3.1.3. Dépendances

Pour tous les tests d'intégration, les tests unitaire doivent être faits avant et doivent être fonctionnels.

3.1.4. Procédure de test

-méthode de départ : on teste si il récupère bien le palet et qu'il l'amène bien au but adverse ensuite qu'il le pose la bas. Cette méthode est une constante de notre début de partie, il faut juste vérifier que les valeurs rentrer en paramètres soient correctes.

- Le but du test pour la méthode recherchePalet() est de trouver le nombre de valeurs que le robot doit prendre en tournant pour qu'il en ait assez souvent pour ne pas louper un palet. Puis on regarde la différence entre les différentes valeurs enregistrées, un écart assez conséquent sera considéré comme un palet.
Après avoir fait son tour, le robot doit pouvoir se remettre en face du palet pour aller le récupérer.

-La méthode prendrePalet() doit tout en avançant vérifier que la distance mesurée est bien inférieur à celle que l'on prend grâce à l'ultraSonicSensor, sinon cela voudrait dire que le palet a disparu, l'autre robot serait partie avec.

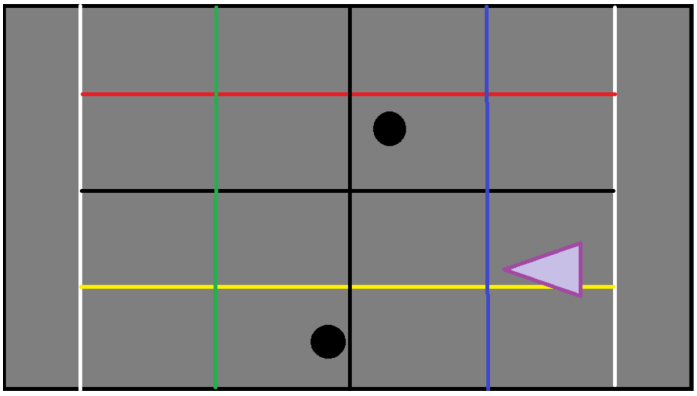
-ramenerPalet() le test consiste à tester le modulo 342 (pour qu'il soit bien en face des cages et que la distance face aux cages diminue toujours, puis quand il passe la ligne blanche il lâche le palet, recule et fait un demi tour.

Tous les tests ont été réalisés sur le terrain de jeu.

4. Annexes

-tableau de valeur qu'il prend quand il tourne:

137 ,137 ,137 ,137 ,137 ,137 ,137 ,137 ,137 ,137 ,135 ,135 ,135 ,135 ,134 ,134 ,134 ,95 ,95 ,95 ,91 ,91 ,91 ,90 ,90 ,90 ,90 ,90 ,90 ,90 ,128 ,128 ,128 ,88 ,88 ,88 ,89 ,89 ,89 ,90 ,90 ,90 ,90 ,90 ,90 ,95 ,95 ,95 ,95 ,85 ,85 ,85 ,87 ,87 ,87 ,111 ,111 ,111 ,112 ,112 ,112 ,112 ,84 ,84 ,84 ,82 ,82 ,82 ,84 ,84 ,84 ,82 ,82 ,82 ,85 ,85 ,85 ,85 ,85 ,85 ,85 ,85 ,85 ,85 ,85 ,85 ,85 ,85 ,85 ,85 ,85 ,77 ,77 ,77 ,77 ,2147483647 ,244 ,244 ,244 ,244 ,244 ,2147483647 ,2147483647 ,2147483647 ,2147483647 ,2147483647 ,2147483647 ,2147483647 ,2147483647 ,2147483647 ,199 ,199 ,199 ,199 ,199 ,199 ,199 ,194 ,194 ,194 ,194 ,192 ,192 ,192 ,192 ,192 ,192 ,192 ,192 ,192 ,192 ,192 ,192 ,192 ,192 ,191 ,191 ,191 ,191 ,193 ,193 ,193 ,193 ,155 ,155 ,155 ,138 ,138 ,138 ,138 ,140 ,140 ,140 ,140 ,135 ,135 ,135 ,134 ,134 ,134 ,134 ,134 ,134 ,134 ,134 ,134 ,134 ,134 ,134 ,134 ,134 ,134 ,134 ,134 ,134 ,134 ,134 ,NB DE VALEUR200



légende :

⚫ Palet

🔽 Pablo

Tableau de valeur pour quand il avance

Méthodes	Réel	Dans le programme
tourner()	360°	322
	90°	80.5
	180°	161
avancer()	9.8cm	100
	29.4cm	300
	19.9cm	200

	49.4cm	500