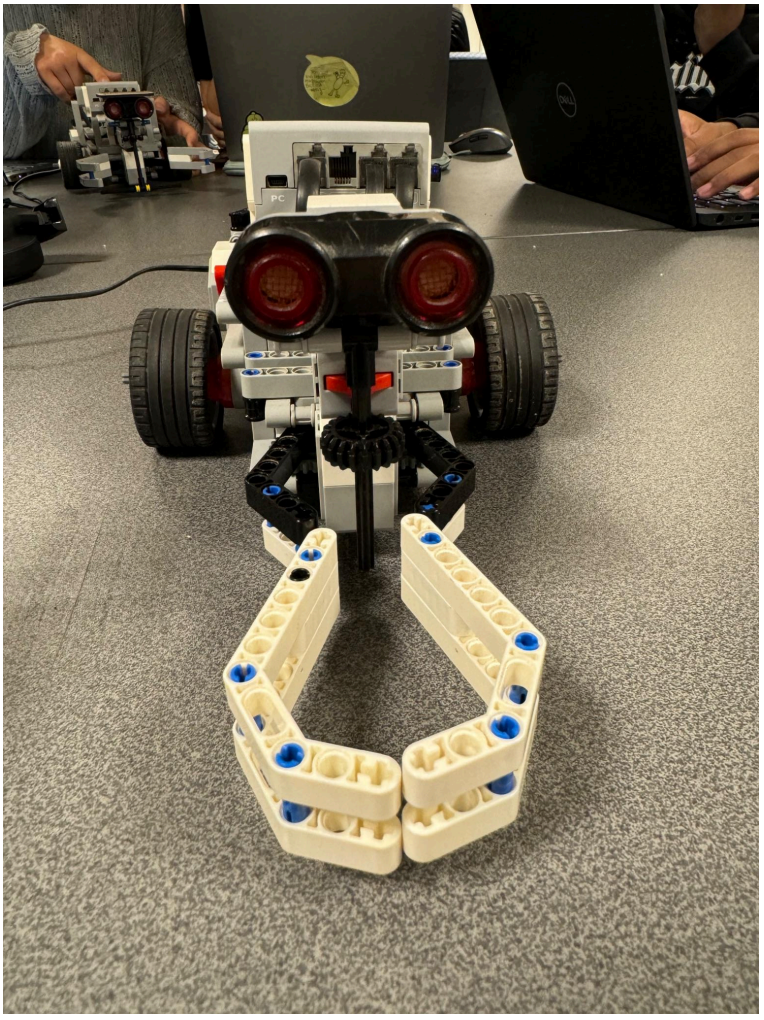
Université Grenoble Alpes

IA 2024-2025

PLANS DES TESTS



Mathias Devilliers n°12201983

Carole Mitton n°11906540

Paul Ndong n°12120816

Saliha Ozturk n°12012637

Université Grenoble Alpes

IA 2024-2025

Sommaire

1. Introduction

Ce document présente le plan de tests élaboré dans le cadre du projet de robotique de notre cours d'intelligence artificielle en L3. Ce projet a pour objectif de programmer un robot capable de récolter un maximum de palets sur le plateau le plus rapidement possible.

L'objectif du plan de tests est de définir les méthodes et critères nécessaires pour valider le bon fonctionnement du robot a chaque étape de son développement. Ce document résume les différents tests à effectuer et leurs procédures pour optimiser et vérifier les performances du robot.

1.1. Objectifs et méthodes

Les tests définis permettront de vérifier que les modules développés respectent les spécifications fonctionnelles et techniques. Ainsi d'assurer que les modules s'intègrent correctement entre eux. Enfin, d'évaluer les performances globales du robot dans différents scénarios de fonctionnement.

Les méthodes utilisées incluront des tests unitaires pour chaque classe, des tests d'intégration pour vérifier les interfaces entre les modules, et des tests fonctionnels pour évaluer le comportement global du robot dans des scénarios réels.

1.2. Document de référence

Pour structurer et rédiger ce plan de tests, nous nous sommes appuyés sur deux principales sources de référence :

Le document proposé par **Damien Pellier**, fournit une méthodologie claire et détaillée pour structurer un plan de tests dans le cadre du projet utilisant des robots LEGO EV3. Ce modèle sert de guide pour organiser les tests en étapes cohérentes, avec une logique d'évaluation progressive.

teaching : ia : project_lego [Damien Pellier Full Professor Univ. Grenoble Alpes]. (s. d.). https://lig-membres.imag.fr/PPerso/membres/pellier/doku.php?id=teaching:ia:project_lego

Suricate. (s. d.). *Plan de test logiciel - le guide ultime*. https://blog.mrsuricate.com/plan-de-test-logiciel-guide-ultime

2. Concepts de base

Le plan de tests repose sur les concepts suivants :

Tests unitaires : vérification de chaque module indépendamment des autres pour s'assurer que les fonctions et méthodes respectent les spécifications.

Tests d'intégration : vérification des interactions entre les modules pour garantir qu'ils fonctionnent ensemble correctement.

Critères de validation : comparaison des résultats obtenus avec les résultats attendus définis dans les spécifications.

3. Tests d'intégration

3.1 Pour chaque test d'intégration

3.1.1 Identifications

Classe Position
Test P.5 : updatePosition
Test P.10 : plusPetitAngleAuRobot

Classe Mouvements
Test M.8 : avancerWhileIsNotPressed
Test M.9 : recherche
Test M.10 : reOrientationMur
Test M.11 : allerChezAdversaire
Test M.13 : allerAuCentre
Test M.14 : avancerVigilantAllerAuCamp
Test M.15 : angleDeRechercheOptimise
Test M.16: avancerJusquaCouleur
Test M.22 : tourneOptimise
Test M.23 : decoupeValeursStrictes
Test M.24 : ajustement

Classe Robot
Test R.1 : premierPalet
Test R.2 : boucleRecherche

3.1.2 Description

Classe Position

Test P.5 : Nous faisons avancer le robot d'une certaine distance avec une certaine orientation puis nous récupérons les valeurs de position en x et y du robot avec les méthode getX et getY en comparant avec les mesures que l'on fait sur le terrain. La méthode updatePosition est appelée dans la méthode avancerDe qui permet au robot d'avancer tout droit.
Test P.10 : Cette méthode permet, en donnant deux points en paramètre, de calculer quelle point possède le plus petit angle avec le robot et de renvoyer celui-ci. Pour tester, on place le robot sur la table en lui donnant deux points et on regarde si, en le faisant tourner de l'angle renvoyé par la méthode, il se tourne vers le point dont l'angle est le plus petit.

Classe Mouvements

Test M.8 : Lancer la méthode avec une distance en décimètres, et vérifier que le robot avance tant que la distance n'est pas atteinte ou que le capteur de toucher n'est pas activé.
Test M.9 : Lancer la méthode avec un angle en degrés et vérifier que le robot tourne correctement de l'angle précisé et qu'il avance vers l'objet le plus proche de lui.
Test M.10 : Cette méthode permet au robot quand il arrive proche du mur du camp adverse de se replacer tout droit devant le mur pour mettre à jour l'orientation du robot. Pour tester nous lançons la méthode et nous regardons s' il se replace bien pike en face du mur.
Test M.11 : Cette méthode permet au robot d'aller dans le camp adverse, de mettre à jour l'orientation grâce à la méthode M.12 et de se retourner pour faire face au terrain de jeu. On initialise le robot à une certaine position sur le terrain et on lance la méthode.
Test M.13 : Cette méthode permet au robot de retourner au centre du terrain en fonction de sa position et de son orientation. Pour tester on lance simplement la méthode avec différente position et orientation du robot sur le terrain.
Test M.14 : Cette méthode permet au robot de prendre d'avancer et de s'arrêter si il rencontre un objet sur son chemin. Il s'arrête pendant 4 secondes si après ce temps l'obstacle est toujours là alors il change de trajectoire. On lance cette méthode et on observe si le robot se rend au camp et son comportement quand on lui présente un obstacle.
Test M.15 : Cette méthode permet au robot, en fonction de sa position, de s'orienter vers le centre du terrain pour avoir plus de chance de trouver des palet et de renvoyer l'angle duquel il va faire une recherche. Pour tester, on lance la méthode en plaçant et initialisant les coordonnées du robot à différent endroit du terrain puis on observe la direction dans laquelle il se tourne ainsi que la valeur qu'il renvoie.
Test M.16: Cette méthode permet au robot d'avancer jusqu'à détecter la couleur mise en paramètre. Pour la tester on lance la méthode et on nous regarde s' il s'arrête bien sur la ligne de la couleur mise en paramètre.
Test M.22 : Cette méthode permet de tourner de manière optimisée vers un angle. Si le robot doit tourner à gauche pour être plus optimisé, alors il tourne à gauche, sinon à droite.
Test M.23 : Cette méthode permet de tourner et de chercher les distances autour de lui. Les distances sont rangées dans une listes puis découpées en fonction de la tendance (croissant ou décroissant).

Test M.24 : Lorsque le robot détecte un palet, cette méthode permet de regarder autour du palet pour se centrer correctement en face du palet. Cela permet de limiter les erreurs dû au mouvement du robot.

Classe Robot

Test R.1 : Cette méthode permet d'aller récupérer le premier palet, celui en face du robot au départ de la compétition. On lance plusieurs fois la méthode pour vérifier qu'elle récupère le palet puis l'emmène au camp adverse avant de faire demi-tour.
Test R.2 : Cette méthode permet de créer la boucles de recherche pour que le robot puisse rechercher et rapporter les palets dans le camp jusqu'à la fin de l'épreuve. Quand on lance cette méthode, le robot va chercher le premier palet puis boucle tant qu'il n'a pas récupéré 9 palets. La boucle fait qu'il recherche deux fois et que s' il ne trouve rien pendant ces deux recherches il va vers le milieu pour rechercher là bas.

3.1.3 Contraintes

Classe Position

Test P.5 : Besoin d'un terrain avec des repères géométriques, besoin de prendre les mesures de la position du robot sur le terrain.
Test P. 10 : Être sur un terrain où l'on connaît les distances pour placer des marqueurs pour les points. Pas de contraintes humaines.

Classe Mouvements

Test M.8 : Aucun obstacle ne doit être posé sur le chemin du robot, aucune intervention humaine.
Test M.9 : Aucun obstacle ne doit être posé autour du robot, afin de permettre sa rotation, aucune intervention humaine.
Test M.10 : Besoin d'être en face d'un mur, pas d'intervention humaine.
Test M.11 : Besoin d'être sur le terrain de jeu, pas d'intervention humaine.
Test M.13 : Besoin d'être sur le terrain de jeu, pas d'intervention humaine.
Test M.14 : Besoin d'intervention humaine pour présenter un obstacle devant le robot.
Test M.15 : Besoin d'être sur le terrain de jeu, pas d'intervention humaine.
Test M.16 : Doit être réalisé dans les mêmes conditions que lors du calibrage du capteur de couleur.
Test M.22 : Besoin d'être sur le terrain, aucune intervention humaine.
Test M.23 : Besoin d'être sur le terrain, aucune intervention humaine.
Test M.24 : Pas de contrainte, pas d'intervention humaine.

Classe Robot

Test R.1 : A tester sur la table avec un palet. Pas besoin d'intervention humaine spécifique
Test R.2 : A tester sur la table avec tous les palets à leur place. Pas besoin d'intervention humaine spécifique.

3.1.4 Dépendances

Classe Position

Test P.5 : Dépendance avec les méthode M.1, M.2, P.1 et P.6
Test P.10 : Dépendance avec P.6 et P.9

Classe Mouvements

Test M.8 : Dépendance avec C.4, M.1, M.4, C.1, M.3.
Test M.9 : Dépendance avec M.2, C.4, M.8, M.4, C.1, M.1.
Test M.10 : Dépendance avec M.2, M.4, C.4 et P.3.
Test M.11 : Dépendance avec M.12, M.1, M.2, M.3.
Test M.13 : Dépendance avec P.6, P.5, P.4, P.3, M.1, M.2.
Test M.14 : Dépendance avec C.3, M.1, M.2, M.6,M.16, P.3
Test M.15 : Dépendance avec P.1, P.2, M.2
Test M.16 : Dépendance avec C.2,C.5
Test M.22 : Dépendance avec M.2.
Test M.23: Dépendance avec M.18.
Test M.24 : Dépendance avec M.2, M.4, C.3, M.8

Classe Robot

Test R.1 : Dépendance avec les méthodes M.3, M.9, M.3, M.2, M.1, M.12
Test R.2 : Dépendance avec les méthodes R.1, M.9, M.11, M.13, M.15.

3.1.5 Procédure de test

Classe Position

Test P.5 : En entrée, un angle pour faire tourner le robot et une distance pour qu'il avance sur le terrain. Si ,après le déplacement du robot, on trouve avec les méthodes getX et getY les mêmes valeurs que celle que l'on peut mesurer sur le terrain alors la méthode est valide.
Test P.10 : En entrée, les coordonnées de deux points. En sortie, elle doit renvoyer l'angle entre le robot et l'un des points qui est le plus petit. On place sur le terrain des marqueurs au même coordonnées que ce données en paramètre et on code le robot pour qu'il tourne de la distance renvoyée par cette méthode. Si le robot s'oriente 5 fois d'affilée alors que l'on change la place des points alors la méthode est validée.

Classe Mouvements

Test M.10 : Pas de valeur en entrée, pas de valeur en sortie. Pour valider le test, le robot doit s'être correctement replacé devant le mur bien en face.
Test M.11 : Pas de valeur en entrée, pas de valeur en sortie. Pour valider le test, le robot doit s'être correctement rendu dans la terrain adverse, s'être réorienté et avec fait un demi-tour.

Test M.13 : Pas de valeur d'entrée ni de sortie. On lance plusieurs fois la méthode en initialisant le robot à différentes positions et orientation du robot. SI le robot va au centre du terrain alors la méthode est validée.
Test M.14 : Pas de valeur en entrée. On lance le robot avec cette méthode est on teste s'il repart quand on retire l'objet avant 4 secondes, s'il change de trajectoire quand l'objet reste plus de 4 secondes et s'il vas jusqu'au camp en s'arrêtant à la ligne blanche s'il n'y a pas d'obstacle. On teste chacune des conditions 3 fois pour que le test soit validé.
Test M.15 : En entrée on initialise la position du robot et on le place sur la table au coordonnées correspondante. On lance la méthode avec différente position de départ et on vérifie que l'angle duquel tourne le robot le place bien dans un direction qui permet de voir les palets vers le centre du terrain.
Test M.22 : En entrée un entier correspondant à l'angle duquel le robot a tourné au total lors de sa recherche, un autre entier pour l'angle duquel le robot a tourné pour atteindre sa cible, et un float correspondant à la distance vue de l'objet vers lequel le robot veut s'orienter.
Test M.23 : En entrée un entier correspondant à l'angle duquel le robot doit tourner pour récupérer les distances. Pour valider ce test, le robot doit retourner une liste de listes contenant toutes les distances vues triées selon leur tendance (distance croissante vs décroissante). Une tolérance de 5 cm de différence est acceptée.
Test M.24 : Pas de données en entrée. On lance une recherche avec le robot en plaçant un palet sur le terrain. On vérifie que le robot se réoriente correctement en face du palet. Si c'est le cas sur 3 essaie d'affilée alors la méthode est validée.

Classe Robot

Test R.1 : On teste plusieurs fois la méthode sur le terrain. Si la méthode fait ce que l'on attend 3 fois de suite sans faute alors elle est validée.
Test R.2 : En entrée, l'emplacement du robot sur le terrain. On lance la méthode plusieurs fois avec le robot sur la table et tous les palets pour vérifier que le robot fait ce qu'on avait prévu. Si le robot réagit comme on le voulait avec différents emplacement de départ alors la méthode est validée.

4. Tests unitaires

4.1 Pour chaque test unitaire

4.1.1 Identification

Classe Capteurs

Test C.1 : isPressed
Test C.2 : capteurDeCouleur
Test C.3 : regarde
Test C.4 : fermeLesYeux
Test C.5: convertiseurCouleur

Classe Position

Test P.1 : getX et getY
Test P.2 : getDegres
Test P.3 : setX, setY et setDegres
Test P.4 : updateOrientation
Test P.6 : calculerPositionPoint
Test P.7 : procheDuCentre
Test P.8 : degesAuCampAdverse
Test P.9 : angleEntreDeuxPoint

Classe Mouvements

Test M.1 : avancerDe, avancerDeRapide
Test M.2 : tournerDe, tournerDeRapide
Test M.3 : ouvreBras et fermeBras
Test M.4 : isMoving
Test M.5 : stop
Test M.6 : delay
Test M.7 : distanceDiminue
Test M.12 : paletValide
Test M.15 : nettoyageDesValeursAbsurdes
Test M.16 : testTempsRotation
Test M.17 : supprimerDoublonsSuccessifs
Test M.18 : rechercheDistances
Test M.19 : supprime
Test M.20 : calculeDistRestante
Test M.21 : minAngle
Test M.25 : premierNestPasUneDisc

Classe Robot

4.1.2 Description

Classe Capteurs

Test C.1 : Lancer la méthode avec un affichage de la sortie de la méthode qui est un booléen pour vérifier que le robot renvoie true quand nous appuyons sur le capteur de pression.
Test C.2 : Lancer la méthode avec un affichage de sa sortie et vérifier que le capteur renvoie les bonnes couleurs malgré les variations légères des valeurs pour chaque couleur en fonction de la luminosité.
Test C.3 : Lancer la méthode avec un affichage de la sortie de la méthode qui est un tableau de décimale qui correspond à la distance entre le robot et un objet placé en face de lui. Nous effectuons plusieurs fois ce test en plaçant l'objet à différente distance pour vérifier les valeurs de sortie.

Test C.4 : Après avoir initié le robot nous lançons la méthode pour vérifier s'il éteint son capteur d'ultrason.
Test C.5 :lancer la méthode avec une couleur en code RGB et vérifier qu'il renvoie bien la couleur sous le capteur.

Classe Position

Test P.1 : Nous initialisons le robot avec des valeurs pour l'attribut x et y, puis nous lançons la méthode pour vérifier qu'il renvoie bien la bonne valeur
Test P.2: Nous initialisons le robot puis nous lui donnons une nouvelle valeur de degrés avec la méthode setDegres. Nous lançons ensuite la méthode getDegres pour voir s' il nous renvoie bien l'orientation du robot.
Test P.3 : Nous initialisons le robot avec des valeurs en X et Y (les degrés sont par défaut à 0). Nous modifions les valeurs avec les méthodes set puis avec les méthodes get nous vérifions que ce sont les nouvelles valeurs.
Test P.4 : Nous initialiser le robot, puis nous le faisons tourner d'un certain angle en degrès. Ensuite nous regardons la valeur que l'on a avec la méthode getDegres et on la compare avec le mesure dans la réalité. On refait ce test avec différents angles auquel le robot tourne au début.
Test P.6 : Cette méthode permet de calculer les coordonnées en x et y d'un point qui serait à une certaine distance du point 0,0 en suivant une orientation donnée. Pour tester nous prenons une distance et une orientation, nous lançons le programme avec un affichage du résultat et nous comparons avec les résultats que nous avons nous même calculé.
Test P.7 : Cette méthode renvoie true si les valeurs dans le tableau en paramètre qui corresponde au coordonnée en x et y d'un point sur le terrain sont à plus ou moins 2 centimètres du centre. Nous lançons la méthode avec un affichage de la sortie en testant différentes valeurs.
Test P.8 : Cette méthode renvoie l'angle dont le robot doit tourner pour ce retrouver face au camp adverse. On lance le robot en lui faisant prendre une orientation aléatoire puis on le fait tourner de la distance renvoyé par l'appel de la méthode. On teste ça avec différente orientation avec l'appel de la méthode.
Test P.9 : Cette méthode renvoie l'angle entre deux points qui sont sur un même plan. Pour tester, on lui donne différentes coordonnées de paires de point et on regarde avec nos calculs si on trouve les mêmes résultats.

Classe Mouvements

Test M.1 : Lancer la méthode avec un entier correspondant à la distance (décimètres) et un booléen correspondant à l'asynchronie pour vérifier la distance parcourue par le robot. La méthode avancerDeRapide à une vitesse plus élevée que la méthode avancerDe, le test est le même.
Test M.2 : Lancer la méthode avec un entier correspondant à l'angle (degré) et un booléen correspondant à l'asynchronie pour vérifier l'angle effectivement tourné par le robot. La méthode tournerDeRapide à une vitesse de rotation plus élevée que la méthode tournerDe, le test est le même.

Test M.3 : Lancer la méthode et vérifier que le robot ouvre/ferme les bras (en fonction de sa situation)
Test M.4 : Lancer la méthode lorsque le robot avance ou tourne. Vérifier que le retour est “true” ou “false” en fonction de si le robot bouge ou pas.
Test M.5 : Lancer la méthode pendant que le robot avance ou tourne. Vérifier que le robot arrête ce qu’il est en train de faire.
Test M.6 : Lancer la méthode et vérifier que le robot attend d’une durée de 25ms.
Test M.7 : Lancer la méthode avec un tableau de float. Vérifier que le robot retourne “true” si le premier élément est plus grand égal au deuxième élément.
Test M.12 : Cette méthode permet de savoir si un palet est dans un camp par rapport à la position du robot sur le terrain et la distance à laquelle il a vu le palet. On peut alors savoir si le robot ne doit pas aller le chercher. En paramètre on donnent les données d’un palet et l'orientation du robot.
Test M.15 : Lancer la méthode avec un tableau de float. Vérifier que le tableau retourné est le même que le tableau fourni privé des éléments nuls.
Test M.16 : Lancer la méthode avec un entier correspondant à l’angle (degré) dont le robot devra tourner.
Test M.17 : Lancer la méthode avec un tableau de float. Vérifier que le tableau retourné est le même que le tableau fourni privé des éléments successifs en doublons.
Test M.18 : Lancer la méthode avec un entier correspondant à l’angle dont le robot doit tourner.
Test M.19 : Lancer la méthode avec un tableau de tableau d’entiers et un entier correspondant à l’indice à de l’élément à supprimer dans ce tableau.
Test M.20 : Lorsque le robot avance il doit pouvoir s'arrêter si un obstacle se présente. Quand il s’arrête il renvoie alors la durée depuis qu’il a commencé à avancer pour permettre de calculer la distance qu’il reste à parcourir en fonction du temps qu’il a avancé et de sa vitesse moyenne.
Test M.21 : La méthode permet de chercher le minimum dans un tableau d’entiers ainsi que son indice d’apparition.
Test M.25 : La méthode permet d’affirmer si la première distance regardée stockée dans une liste de listes des distances correspondant à une discontinuité physique ou non.

Classe Robot

4.1.3 Contraintes

Classe Capteurs
Test C.1 : Pas d’environnement particulier, l’un des testeur appuie sur le capteur de pression quand la méthode est lancée pour l’actionner.

Test C.2 : Les couleurs détectées en code RGB sont calibrées pour la salle et la table où se déroule la compétition, dans des conditions de luminosité constantes. Ainsi, le capteur détecte les bonnes couleurs uniquement dans ces conditions. Par exemple, si la lumière était éteinte, les valeurs seraient complètement différentes.
Test C.3 : Un objet doit être placé en face du robot. Nous changeons la distance entre l'objet et le robot plusieurs fois.
Test C.4 : Pas de contrainte spécifique ni d'intervention humaine.
Test C.5 :calibré pour les couleurs de la table de la compétition.

Classe Position

Test P.1 : Pas de contrainte spécifique ni d'intervention humaine.
Test P.2 : Pas de contrainte spécifique ni d'intervention humaine.
Test P.3 : Pas de contrainte spécifique ni d'intervention humaine.
Test P.4 : Pas de contrainte d'environnement, besoin de prendre un mesure de l'orientation dans la réalité
Test P.6 : Pas de contrainte spécifique, besoin de vérifier les calculs à la main.
Test P.7 : Pas de contrainte spécifique ni d'intervention humaine.
Test P.8 : Pas de contrainte spécifique ni d'intervention humaine.
Test P.9 : Pas de contrainte spécifique, besoin de l'intervention d'un humain pour vérifier les résultats.

Classe Mouvements

Test M.1 : Aucun obstacle doit être posé sur le chemin du robot, afin de permettre son mouvement, aucune intervention humaine.
Test M.2 : Aucun obstacle doit être posé autour du robot, afin de permettre sa rotation, aucune intervention humaine.
Test M.3 : Pour ouvrir les bras, l'attribut etatBras doit être "false". Vice-versa pour fermer les bras, aucune intervention humaine.
Test M.4 : Aucune contrainte.
Test M.5 : Aucune contrainte, aucune intervention humaine.
Test M.6 : Aucune contrainte, aucune intervention humaine.
Test M.7 : Aucune contrainte, aucune intervention humaine.
Test M.12 : Besoin d'être sur le terrain de jeu avec un palet, pas d'intervention humaine.
Test M.15 : Aucune contrainte, aucune intervention humaine.
Test M.16 : Besoin d'être sur le terrain de jeu, aucune intervention humaine.
Test M.17 : Aucune contrainte, aucune intervention humaine.
Test M.18 : Besoin d'être sur le terrain de jeu, aucune intervention humaine.
Test M.19 : Aucune contrainte, aucune intervention humaine.
Test M.20 : Aucune contrainte, intervention humaine pour placer un obstacle.
Test M.21 : Aucune contrainte ni intervention humaine nécessaire.
Test M.22 : Aucune contrainte, aucune intervention humaine.
Test M.25 : Besoin d'être sur le terrain de jeu. Le testeur doit mesurer si la première distance à gauche et/ou à droite de la distance en face du robot est supérieure ou inférieure à 5cm.
.

Classe Robot

4.1.4 Dépendances

Classe Capteurs

Test C.1 : Pas de dépendances
Test C.2 : C.5
Test C.3 : Pas de dépendance
Test C.4 : Pas de dépendance
Test C.5 : Pas de dépendances

Classe Position

Test P.1 : Pas de dépendance
Test P.2: Dépendance avec la méthode P.3
Test P.3 : Dépendance avec les méthodes P.3
Test P.4 : Dépendance avec la méthode P.3
Test P.6 : Pas de dépendance.
Test P.7 : Pas de dépendance.
Test P.8 : Dépendance avec la méthode M.2
Test P.9 : Pas de dépendance.

Classe Mouvements

Test M.1 : Dépendance avec P.5
Test M.2 : Dépendance avec P.4
Test M.3 : Aucune dépendance.
Test M.4 : Aucune dépendance.
Test M.5 : Aucune dépendance.
Test M.6 : Aucune dépendance.
Test M.7 : Aucune dépendance.
Test M.12 : Dépendance avec P.1
Test M.15 : Aucune dépendance.
Test M.16 : Aucune dépendance.
Test M.17 : Aucune dépendance.
Test M.18 : Dépendance avec C.3 et M.2.
Test M.19 : Aucune dépendance.
Test M.20 : Aucune dépendance.
Test M.21 : Aucune dépendance.
Test M.25 : Aucune dépendance.

Classe Robot

…

4.1.5 Procédure de test

Classe Capteurs

Test C.1 : Pas de données en entrée, la valeur de sortie doit être true quand le capteur est actionné, false quand il ne l'est pas.
Test C.2: Pas de données en entrée, la valeur de sortie est la couleur détectée par le capteur.
Test C.3 : Les données en entrée au début du test sont un tableau de float vide. Il est attendu que dans ce tableau nous retrouvions la distance entre le robot et l'objet placé en face. Si le robot renvoie une distance proche au centimètre près de la distance réelle alors la méthode est validée.
Test C.4 : Pas de données en entrée. Les résultats attendus sont que lors du lancement de la méthode le robot éteigne son capteur d'ultrason. Si c'est le cas, le test est validé.
Test C.5 : Les données entrées sont le code RGB d'une couleur, la valeur de sortie est la couleur détectée par le capteur.

Classe Position

Test P.1 : En entrée, les valeurs de x et y lors de l'initialisation du robot. Les résultats attendus sont le retour des valeurs x et y qui sont données lors de l'initialisation du robot.
Test P.2 : Pas de données en entrée. Les résultats attendus sont de retrouver la même valeur que celle qui lui aura été donnée lors de l'appel à la méthode setDegres. Si c'est le cas, la méthode est validée.
Test P.3 : Les données en entrées sont les nouvelles données en X, Y et degrés. En sortie nous attendons ces mêmes valeurs. Si les valeurs que l'on a accorder à X, Y et degrés sont les mêmes que celles que l'on retrouve en utilisant les méthodes getX, getY et getDegres alors les méthodes sont validées.
Test P.4 : En entrée, une valeur pour l'angle duquel doit tourner le robot au début. Si le robot renvoie avec la méthode getDegres la même valeur que celle observé avec le mesure du robot dans l'environnement alors la méthode est validée.
Test P.6 : En entrée, une distance et un angle. On va ensuite comparer les résultats de la méthode avec les résultats aux calculs de trigonométrie que nous faisons. Si les résultats sont identiques alors la méthode est validée.
Test P.7 : En entrée, un tableau avec deux valeurs : la coordonnée en x et en y d'un point. Si la méthode ne renvoie true que pour les valeurs qui sont à plus ou moins 2 centimètre du centre alors elle est validé.
Test P.8 : Au départ on fait tourner le robot d'angle variable. Ensuite on regarde si après avoir fait tourner le robot de la distance renvoyer par la méthode degresAuCamp il se place face au camp adverse. Si c'est le cas alors la méthode est validée.
Test P.9 : En entrée, les coordonnées de deux points. Si la méthode renvoie les mêmes résultats que ce que l'on trouve en faisant les calculs à la calculatrice alors la méthode est validée.

Classe Mouvements

Test M.1 : Une distance en entrée (décimètres), pas de valeur en sortie. Pour valider le test, la distance entre la position du robot avant le test et après le test doit correspondre à la distance mise en paramètre.
Test M.2 : un angle en entrée, pas de valeur en sortie. Pour valider le test, le robot doit avoir tourné d'autant d'angles que passé en paramètre après le test.
Test M.3 : Pas de valeur en entrée, pas de valeur en sortie. Pour valider le test, le robot doit fermer ses bras (resp. ouvrir ses bras) si ses bras sont ouverts (resp. fermés).
Test M.4 : Pas de valeur en entrée. Pour valider le test, le robot doit retourner "true" s'il est en mouvement, "false" sinon.
Test M.5 : Pas de valeurs en entrée, pas de valeur en sortie. Pour valider ce test, le robot doit s'arrêter d'être en mouvement s' il est en mouvement.
Test M.6 : Pas de valeurs en entrée, pas de valeur en sortie. Pour valider ce test, le robot doit attendre 25ms avant de pouvoir continuer à effectuer autre chose. (En pratique, il faut augmenter l'attribut "DELAY" pour tester le temps effectif).
Test M.7 : Le programme doit avoir un tableau de float en entrée simulant des distances. Pour valider ce test, le programme doit retourner true si la première valeur est supérieure ou égale à la deuxième, false sinon.
Test M.8 : Une distance doit être fournie en entrée (décimètres). Pas de valeur en sortie. Pour valider ce test, le robot doit avancer de la distance mise en paramètres. Si le capteur tactile est activé en route, le robot doit s'arrêter.
Test M.9 : Un angle doit être fourni en entrée (degrés). La liste des discontinuités est renvoyée en retour. Pour valider ce test, le robot doit tourner de l'angle précisé et doit s'avancer vers l'objet le plus proche de lui.
Test M.12 : En entrée dans un tableau la distance du robot à un palet et l'orientation à laquelle il a été vue. Si le robot renvoie true quand les coordonnée indique des palet qui ne sont pas dans les camps et false à l'inverse alors la méthode est validé.
Test M.15 : Un tableau de float en entrée. Pour valider le test, le robot doit retourner la même tableau privé de ses éléments nuls.
Test M.16 : Un entier correspondant à l'angle à tourner en entrée. Pour valider ce test, le robot doit afficher le temps effectivement pris pour effectuer la rotation.
Test M.17 : Un tableau de float en entrée. Pour valider le test, le robot doit retourner le même tableau privé de ses éléments successifs en doublons.
Test M.18 : Un entier en entrée correspondant à l'angle dont le robot doit tourner. Pour valider le test, le robot doit effectivement tourner du bon angle, et doit retourner un tableau de float ayant les distances vues par le robot durant sa rotation.
Test M.19 : Un tableau de tableaux entiers en entrée et un entier correspondant à l'indice de l'élément à supprimer. Pour valider ce test, la méthode doit retourner un tableau de tableaux entiers ayant tous les éléments du tableau d'origine privé de l'élément d'indice souhaité supprimé.
Test M.20 : En entrée, la distance dont le robot va avancer. On place à une distance donnée un objet et on voit si la méthode renvoie la distance qu'il restait à parcourir en mesurant. Si les valeurs sont identiques, le test est validé.

Test M.21 : Un tableau d'entiers en entrée. Pour valider ce test, la méthode doit retourner un tableau deux éléments correspondant respectivement à l'indice minimum ainsi que son indice d'apparition.
Test M.25 : Une liste de listes de float correspondant aux distances rangées et clusterisées. Pour valider la méthode, le robot doit retourner true si la distance vue au début est bien une discontinuité et false sinon. Pour savoir si une distance est une discontinuité, elle doit être différente à au moins 5cm de la première distance à gauche d'elle-même ou à droite d'elle-même.

Classe Robot

5.Glossaire

Asynchrone : Mode de fonctionnement où le robot exécute une tâche sans attendre qu'elle soit terminée avant de passer à une autre.
Camp adverse : Zone située en face de la position de départ du robot sur le terrain.
Palet valide : Palet détecté et récupérable selon les conditions définies dans le cahier des charges.
LeJOS : Framework Java utilisé pour programmer les robots LEGO EV3.

6.Références

leJOS EV3 API documentation. (s. d.). https://lejos.sourceforge.io/ev3/docs/

7.Index