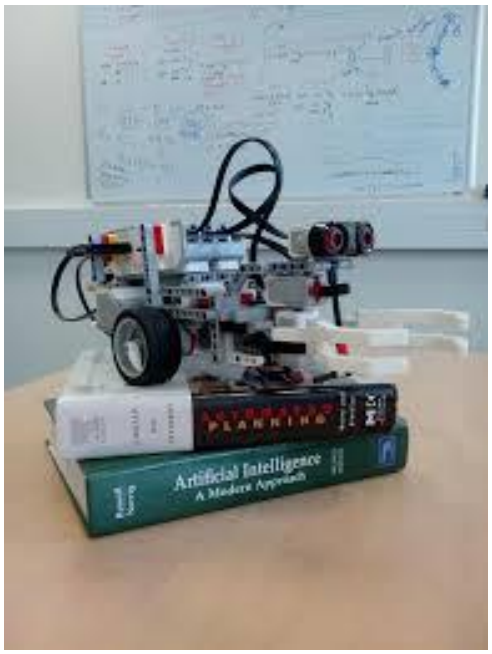
Université Grenoble Alpes

Plan de test



Projet Toby

Plan de test

Projet Toby

Les informations d'identification du document

Référence du document :	D7
Version du document :	1.01
Date du document :	14/12/2020
Auteurs :	FLEURY Pierre JORDAN Célia JULIARD Victor KOSAREVA Margarita

Les éléments de vérification du document

Validé par :	FLEURY Pierre JORDAN Célia JULIARD Victor
Validé le :	14/12/2020
Soumis le :	14/12/2020
Type de diffusion :	Document électronique (.pdf)
Confidentialité :	Standard/Etudiants licence MIASHS à Grenoble

Mots clés : plan de test

Table des matières

1. Introduction

1.1 Objectifs et méthodes

Le but de ce projet est de crée un code qui permette au robot de récupérer un maximum de palet sur un terrain défini (3m*2m) et de les ramener dans les « cages ». Pour cela il faut que le robot puisse maitriser plusieurs méthodes et qu'il puisse repérer tout seul les différents palets sur le plateau. Les palets sont normalement placer à un endroit fixe mais ils peuvent se retrouver à être déplacer, le robot doit donc pouvoir les détecter peu importe leur position.

2. Concepts de base

Afin de comprendre sans difficulté le document voici quelques définitions :

Int	Type d'un objet qui doit être un nombre entier
Float	Type d'un objet qui doit être un nombre entier ou réel
String	Type d'un objet qui doit être une chaine de caractère
Char	Type d'un objet qui doit être un seul caractère

3. Test fonctionnels

3.1 Pour chaque scénario

3.1.1 Identification

Chaque test est répété plusieurs fois afin de les vérifier au mieux.

Méthodes	**Test 1**	**Test 2**	**Test 3**
Avancer (int v, int d)	1.01	1.02	1.03
Reculer (int v, int d)	1.04	1.05	1.06
Orienter (int a)	1.07	1.08	1.09
Rechercher ()	1.10	1.11	1.12
getDistance ()	1.13	1.14	1.15
ouvrirPinces()	1.16	1.17	1.18
fermerPinces()	1.19	1.20	1.21
CapteurCouleur()	1.22	1.23	1.24
isPressed()	1.25	1.26	1.27

3.1.2 Description

1.01	On test la méthode **avancer** sur **30cm** avec une vitesse de **500mm/sec** sur un espace plat.
1.02	On test la méthode **avancer** sur **1m** avec une vitesse de **500mm/sec** sur un espace plat.

1.03	On test la méthode **avancer** sur **1m** avec une vitesse de **400mm/sec** sur un espace plat.
1.04	On test la méthode **reculer** sur **30cm** avec une vitesse de **500mm/sec** sur un espace plat.
1.05	On test la méthode **reculer** sur **1m** avec une vitesse de **500mm/sec** sur un espace plat.
1.06	On test la méthode **reculer** sur **1m** avec une vitesse de **40mm/sec** sur un espace plat.
1.07	On test la méthode **orienter** de **45°** avec une vitesse de **100mm/sec** sur un espace plat.
1.08	On test la méthode **orienter** de **90°** avec une vitesse de **100mm/sec** sur un espace plat.
1.09	On test la méthode **orienter** de **180°** avec une vitesse de **100mm/sec** sur un espace plat.
1.10	On test la méthode **rechercher** avec le **palet à droite.**
1.11	On test la méthode **rechercher** avec le **palet à gauche**.
1.12	On test la méthode **rechercher** avec le **palet derrière.**
1.13	On test la méthode **getDistance()** avec une distance de **30cm**.
1.14	On test la méthode **getDistance()** avec une distance de **50cm**.
1.15	On test la méthode **getDistance()** avec une distance de **1m**.
1.16	On test la méthode **ouvrirPinces()**avec la **vitesse maximum** possible pendant **100 secondes**
1.17	On test la méthode **ouvrirPinces()**avec la **vitesse maximum** possible pendant **400 secondes**
1.18	On test la méthode **ouvrirPinces()**avec la **vitesse maximum** possible pendant **800 secondes**
1.19	On test la méthode **fermerPinces()** avec la **vitesse maximum** possible pendant **100 secondes**
1.20	On test la méthode **fermerPinces()** avec la **vitesse maximum** possible pendant **400 secondes**
1.21	On test la méthode **fermerPinces()**avec la **vitesse maximum** possible pendant **800 secondes**
1.22	On test la méthode **CapteurCouleur()** avec du rouge, sur un espace plat.
1.23	On test la méthode **CapteurCouleur()** avec du bleu, sur un espace plat.
1.24	On test la méthode **CapteurCouleur()** avec du blanc, sur un espace plat.
1.25	On test la méthode **isPressed()** en enclenchant le capteur toucher à la main pendant quelques secondes
1.26	On test la méthode **isPressed()** en faisant avancer le robot de 1m afin que le capteur soit enclencher par le palet qui lui est à 30cm
1.27	On test la méthode **isPressed()** en faisant avancer le robot de m afin que le capteur soit enclencher par le palet qui lui est à 30cm

3.1.3 Contraintes

1.01 à 1.09	La vitesse des moteurs ainsi que la synchronisation des roues.

1.10 à 1.12	On ne peut pas faire de distinction entre un palet et un mur a part une fois que l'on est assez proche de celui-ci.
1.13 à 1.15	Comme la distance est détecter par une vision en forme de cône il faut bien faire attention à ce que rien ne soit capter dans les alentours qui serait gênant.
1.16 à 1.21	Il ne faut pas trop ouvrir ou trop fermer les pinces pour éviter de casser le mécanisme.
1.22 à 1.24	La lumière ambiante n'est pas toujours la même ainsi le blanc n'est pas le même selon celle-ci.
1.25 à 1.27	Le capteur toucher ne s'enclenche que s'il est pressé pendant une petite durée ainsi il peut se retrouver à ne pas toujours être presser par le palet.

3.1.4 Dépendances

1.01 à 1.09	La synchronisation des roues
1.10 à 1.12	Les méthodes avancer(), orienter() et getDistance()
1.13 à 1.15	Les méthodes prédéfinis getDistanceMode() et fetchSample([],int)
1.16 à 1.21	Les méthodes prédéfinis rotate(int) et getMaxSpeed()
1.22 à 1.24	Les méthodes prédéfinis setFloodlight(color), fetchSample(color,int)
1.25 à 1.27	La méthode prédéfini fetchSample([],int)

3.1.5 Procédure de test

Test	Résultats attendus	Critère de validation	Résultats obtenus
1.01	On veut que le robot avance droit.	Si le robot avance bien en ligne droite	Le robot avance bien droit.
1.02	On veut que le robot avance droit.	Si le robot avance bien en ligne droite	Le robot n'avance pas bien droit, la vitesse des moteurs est trop rapide
1.03	On veut que le robot avance droit.	Si le robot avance bien en ligne droite	Le robot avance bien droit.
1.04	On veut que le robot recule droit.	Si le robot recule bien en ligne droite	Le robot recule bien droit.
1.05	On veut que le robot recule droit.	Si le robot recule bien en ligne droite	Le robot ne recule pas bien droit, la vitesse

			des moteurs est trop rapide
1.06	On veut que le robot recule droit.	Si le robot recule bien en ligne droite	Le robot recule bien droit.
1.07	On veut que le robot tourne de 45°	Si le robot tourne de 45°	Le robot tourne correctement.
1.08	On veut que le robot tourne de 90°	Si le robot tourne de 90°	Le robot tourne correctement.
1.09	On veut que le robot tourne de 180°	Si le robot tourne de 180°	Le robot tourne correctement.
1.10	On veut que le robot distingue un mur d'un palet	Si le robot arrive à récupérer un palet	Il y arrive bien.
1.11	On veut que le robot distingue un mur d'un palet	Si le robot arrive à récupérer un palet	Il détecte les murs avant le palet et prend donc beaucoup de temps mais fini par le récupérer.
1.12	On veut que le robot distingue un mur d'un palet	Si le robot arrive à récupérer un palet	Il détecte les murs avant le palet et prend donc beaucoup de temps mais fini par le récupérer.
1.13	On veut que le robot nous retourne la distance devant lui	Si le robot retourne la bonne distance	Le robot retourne bien la bonne distance
1.14	On veut que le robot nous retourne la distance devant lui	Si le robot retourne la bonne distance	Le robot retourne bien la bonne distance
1.15	On veut que le robot nous retourne la distance devant lui	Si le robot retourne la bonne distance	Le robot retourne bien la bonne distance
1.16	On veut que le robot ouvre les pinces assez pour récupérer un palet	Si l'ouverture est assez grande pour récupérer un palet	L'ouverture n'est pas assez grande
1.17	On veut que le robot ouvre les	Si l'ouverture est assez grande	L'ouverture est assez grande

	pinces assez pour récupérer un palet	pour récupérer un palet	
1.18	On veut que le robot ouvre les pinces assez pour récupérer un palet	Si l'ouverture est assez grande pour récupérer un palet	L'ouverture est trop grande
1.19	On veut que le robot ferme les pinces assez pour récupérer un palet	Si l'ouverture est assez petite pour récupérer un palet	La fermeture n'est pas assez petite
1.20	On veut que le robot ferme les pinces assez pour récupérer un palet	Si l'ouverture est assez petite pour récupérer un palet	La fermeture tient bien le palet
1.21	On veut que le robot ferme les pinces assez pour récupérer un palet	Si l'ouverture est assez petite pour récupérer un palet	La fermeture est trop petite cela décale les pinces
1.22	On veut voir s'il ne détecter pas du blanc alors qu'il y a du rouge	S'il ne capte rien	Il ne détecte rien
1.23	On veut voir s'il ne détecter pas du blanc alors qu'il y a du rouge	S'il ne capte rien	Il ne détecte rien
1.24	On veut voir s'il détecte du blanc	S'il capte du blanc	Il ne détecte rien ce qui ne va pas.
1.25	On veut que le robot détecte un palet si le capteur toucher est enclencher	S'il détecte que le capteur toucher est enclencher	Il le détecte bien
1.26	On veut que le robot détecte un palet si le capteur toucher est enclencher	S'il détecte que le capteur toucher est enclencher	Il le détecte bien
1.27	On veut que le robot détecte un palet si le capteur toucher est enclencher	S'il détecte que le capteur toucher est enclencher	Il ne le détecte pas toujours bien

4. Test d'intégration

4.1 Pour chaque scénario

4.1.1 Identification

Méthodes	Test 1	Test 2	Test 3
getPremierPalet()	2.01	2.02	2.03
deposerPalet()	2.04	2.05	2.06

4.1.2 Description

2.01	On test la méthode **getPremierPalet()** en mettant celui-ci à **30cm** avec une vitesse de **500mm/sec** sur un espace plat.
2.02	On test la méthode **getPremierPalet()** en mettant celui-ci à **50cm** avec une vitesse de **500mm/sec** sur un espace plat.
2.03	On test la méthode **getPremierPalet()** en mettant celui-ci à **1m** avec une vitesse de **500mm/sec** sur un espace plat.
2.04	On test la méthode **deposerPalet()** en mettant celui-ci à **droite de la ligne d'arrivée** avec une vitesse de **500mm/sec** sur un espace plat.
2.05	On test la méthode **deposerPalet()** en mettant celui-ci à **gauche de la ligne d'arrivée** avec une vitesse de **500mm/sec** sur un espace plat.
2.06	On test la méthode **deposerPalet()** en mettant celui-ci à **face de la ligne d'arrivée** avec une vitesse de **500mm/sec** sur un espace plat.

4.1.3 Contraintes

2.01 à 2.03	Ce n'est que du code dur ainsi si le palet n'est pas placer correctement cela ne fonctionnera pas
2.04 à 2.06	Il faut qu'il puisse détecter le mur d'en face à moins d'une certaine distance

4.1.4 Dépendances

2.01 à 2.03	Il faut que les méthodes avancer et orienter fonctionne bien.
2.04 à 2.06	Il faut que les méthodes avancer, getDistance et orienter fonctionne bien

4.1.5 Procédure de test

Test	Résultats attendus	Critère de validation	Résultats obtenus

2.01	On veut que le robot récupère le palet rapidement et qu'il le ramène dans les « cages »	Si le robot ramène le palet au bon endroit	Cela fonctionne bien.
2.02	On veut que le robot récupère le palet rapidement et qu'il le ramène dans les « cages »	Si le robot ramène le palet au bon endroit	Cela fonctionne bien.
2.03	On veut que le robot récupère le palet rapidement et qu'il le ramène dans les « cages »	Si le robot ramène le palet au bon endroit	Cela ne fonctionne pas correctement il ne récupère pas le palet.
2.04	On veut que le robot ramène le palet dans les « cages »	Si le robot ramène le palet au bon endroit	Cela fonctionne bien.
2.05	On veut que le robot ramène le palet dans les « cages »	Si le robot ramène le palet au bon endroit	Cela fonctionne bien.
2.06	On veut que le robot ramène le palet dans les « cages »	Si le robot ramène le palet au bon endroit	Cela fonctionne bien.

5. Tests unitaires

Pour notre projet, aucun test unitaire n'a été nécessaire car cela était plus simple et non indispensable.