L3 MIASHS –
Initiation à l'intelligence artificielle

CAHIER DES CHARGES

ASSIKIDANA Esdras
BOULLI Marouan
GUYOT Alizée
LAURENT-PAOLI Pierre
MEHDI Khadidja

2020

Sommaire :

Tables des illustrations :

1. Introduction

Ce document a pour but de présenter les objectifs, les besoins et les contraintes liés à la réalisation du projet.

2. Rôles

2.1. Maîtrise d'œuvre

M. Damien PELLIER, enseignant chercheur en informatique.

2.2. Maîtrise d'ouvrage

Esdras ASSIKIDANA, Marouan BOULLI, Alizée GUYOT, Khadidja MEHDI et Pierre LAURENT-PAOLI, étudiants en troisième année de licence MIASHS.

3. Contexte

Ce projet s'inscrit dans le cadre d'un enseignement de renforcement en informatique intitulé « Initiation à l'intelligence artificielle », au sein de la troisième année de licence MIASHS à l'Université Grenoble-Alpes.

Cet enseignement a pour but de familiariser les étudiants aux fondements de l'intelligence artificielle à travers la conception et le développement d'un robot LEGO ramasseur de palets.

4. Historique

Les étudiants répartis en groupes participent à une compétition opposant deux robots (un robot par groupe), dont l'objectif est de ramasser et de déposer un maximum de palets dans la zone d'en-but adverse délimitée par une ligne blanche.
Chaque match est composé de deux manches de 5 minutes.
Le robot marquant le plus de points au cours de ces deux manches remporte la victoire.
Ci-dessous, une représentation du terrain sur lequel les robots s'affronteront :

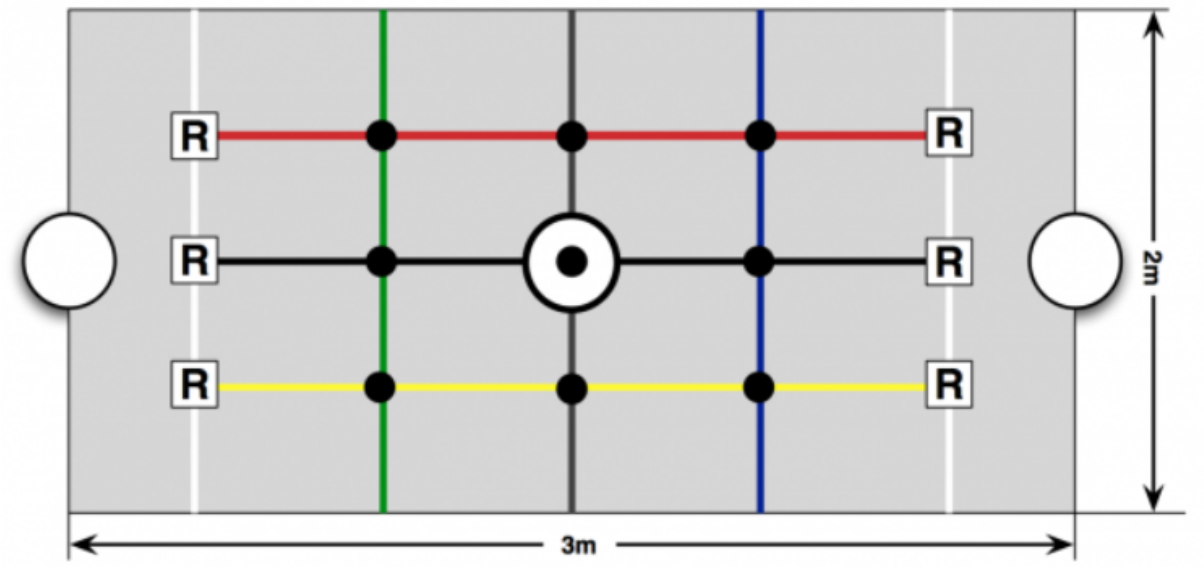

Figure 1: Terrain de jeu

Les R représentent les positions initiales possibles de chacun des robots.

9 palets sont disposés aux intersections des lignes.
Les lignes blanches représentent les lignes d'en-but.

5. Description de la demande

Écrire un code Java embarqué sur le robot qui lui permettra de participer à la compétition[1]. Il faut qu'il joue en autonomie, et dépose un maximum de palets dans la zone d'en-but adverse en un minimum de temps.

Les collisions sont à éviter.

6. Les objectifs

Il faut que le robot remporte un maximum de matchs face aux robots adverses le jour de la compétition. Pour gagner un match, il faut marquer le plus de points possibles en déposant des palets dans le camp adverse. Il y a trois manières de gagner des points :

- Le premier palet déposé par l'une des deux équipes vaut 5 points,
- Les palets suivants marqués valent 3 points,
- Les palets contenus entre les pinces d'un robot mais qui ne sont pas déposés avant la fin du match valent 2 points.

Pour cela, le code Java doit lui permettre de mettre en place une ou plusieurs stratégies, les plus efficaces possible, pour remporter la victoire.

7. Produit du projet

Code Java embarqué sur un robot LEGO, selon la librairie LeJOS[2], avec plusieurs classes permettant de représenter le robot et ses différentes composantes (capteurs d'ultrasons, de toucher, de couleurs, moteurs), les états dans lesquels il pourrait se trouver, une représentation du terrain (un circuit avec des points que le robot doit suivre) ainsi que des méthodes permettant d'accéder aux différentes composantes du robot et de réaliser des actions utilisant ces différentes composantes de manière synchronisée. (cf Plan de développement)

8. Les fonctions du produit

8.1 Description des états et actions de l'automate

Le robot réalise différentes actions en fonction de l'état dans lequel il se trouve. Ci-dessous une description des actions réalisées par le robot dans chacun de ces états (le nom des états est le même que celui des attributs dans la classe DB du code).

FirstPointCMD : cet état correspond au démarrage de la partie (ou après un temps mort). Le robot avance en ligne droite tout en ouvrant les pinces pour récupérer le palet se trouvant en face de lui. Une fois le palet touché (par le capteur de toucher), le robot passe dans l'état suivant.

FirstSaisieCMD : ferme les pinces pour saisir le premier palet en début de partie. Une fois le palet saisi, il passe à l'état suivant.

---

[1] **Règlement de la compétition :** https://lig-membres.imag.fr/PPerso/membres/pellier/doku.php?id=teaching:ia:project_lego

[2] **Documentation LeJOS** :
http://www.lejos.org/ev3.php

FirstDirectionCMD : se décale sur la droite ou la gauche (en fonction de la position de départ) pour éviter les palets se trouvant en chemin, puis se remet en direction de la ligne d'en-but adverse.

GoToButCMD : avance en ligne droite jusqu'à détecter la ligne blanche. Une fois la ligne blanche détectée, passe à l'état suivant.

ButCMD : lâche le palet et passe à l'état suivant.

CalibrateCMD : effectue un petit balayage pour détecter la distance la plus proche au mur et effectue une rotation pour se remettre à l'angle correspondant à cette distance. De cette manière, le robot calibre sa position en formant un angle droit avec le mur. Une fois en position calibrée, passe à l'état suivant.

PointCMD : le robot se trouve face au mur derrière la ligne d'en-but adverse. Il effectue une rotation d'un angle correspondant à la direction du point suivant sur le circuit, puis avance d'une distance déterminée de telle sorte que le robot soit suffisamment éloigné du palet situé sur le point en question, pour permettre sa détection avec le capteur d'ultrasons. Une fois cette distance parcourue, passe à l'état suivant.

SearchCMD : effectue un balayage pour vérifier la présence d'un palet proche de ce point (en comparant les distances renvoyées par le capteur d'ultrasons et une distance maximale définie à partir de mesures sur le terrain). Si un palet est trouvé, passe à l'état suivant, sinon repasse dans l'état PointCMD pour se diriger vers le point suivant dans le circuit.

GoToPaletCMD : avance d'une distance légèrement supérieure à celle renvoyée par le capteur d'ultrasons lors de la détection du palet pour le toucher, tout en ouvrant les pinces. Si le palet n'est pas touché, passe à l'état suivant ; sinon, passe à l'état SaisieCMD décrit plus loin.

AfterOpenPinceCMD : avance d'une petite distance au cas où le palet n'aurait pas été touché à cause d'une marge d'erreur sur le calcul de la distance. Si le palet n'est pas touché, passe à l'état suivant ; sinon, passe à l'état SaisieCMD décrit plus loin.

PaletNotTouchedCMD : dans le cas où le palet n'a pas été touché malgré qu'il ait été détecté, le robot retente sa chance. Il recule d'une distance déterminée de telle sorte à ce qu'il soit suffisamment éloigné pour pouvoir détecter le palet. Puis, il passe à l'état SearchCMD décrit précédemment.

SaisieCMD : ferme les pinces pour saisir le palet et passe à l'état suivant.

DirectionButCMD : se remet en direction de la ligne d'en-but adverse et passe à l'état ButCMD décrit précédemment.

La vérification des états du robot et l'exécution des tâches sont effectuées en parallèle grâce aux méthodes start() et run() de la classe Thread.

8.2 Diagramme d'états de l'automate

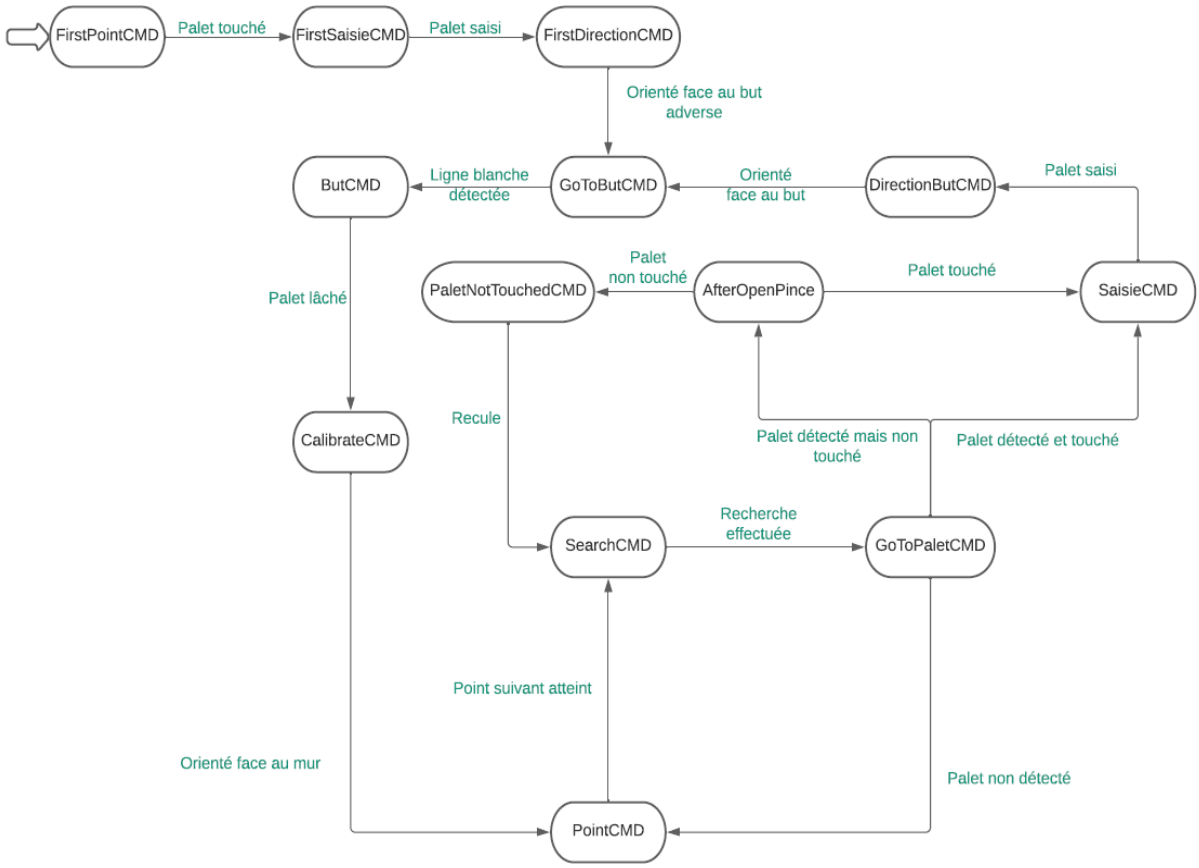

Figure 2: Diagramme d'états

9. Contraintes

9.1. Contraintes de délai :

Les objectifs de notre projet doivent être atteints avant la fin du semestre dans le cadre de l'évaluation de notre UE, mais également compte tenu de la compétition de robots qui aura lieu durant le dernier cours et que nous souhaitons remporter. Un échéancier de documents à rendre a été mis en place par notre professeur et se présente comme ceci :

Semaine	Tâche	Documents à rendre
n°1	Définition des objectifs	
n°2	Analyse des besoins	
n°3	Spécification	Cahier des charges
n°4	Conception	
n°5	Développement	Plan de développement
n°6	Développement	
n°7	Développement	
n°8	Développement	
n°9	Développement	
n°10	Intégration	Plan de tests
n°11	Recette	Code source et documentation interne
n°12	Évaluation	Rapport final

Figure 3: Échéancier de documents à rendre

Étant donné les circonstances sanitaires actuelles, la compétition a été reporté au lundi 30 novembre 2020, et le rendu des documents au lundi 14 décembre 2020.

9.2. Contraintes matérielles :

Notre robot Lego Mindstorms EV3 surnommé « **X AE A-12** » nous a été prêté par le Fablab MSTIC. C'est la brique EV3 qui reçoit le code et contrôle tous les moteurs et capteurs qui y sont branchés. Des pièces sont ajoutées au robot pour d'autres fonctionnalités, les pinces notamment.

Il est accompagné de son chargeur ainsi que d'un palet, identique à ceux utilisés pour la compétition. Nous avons la possibilité de le ramener chez nous ou de le laisser au labo. Nous avons également à disposition le terrain de jeu où se déroulera la compétition.

Pour mener à bien notre projet, nous avons besoin d'au moins un ordinateur doté du logiciel de programmation JAVA Eclipse, ainsi que de la machine virtuelle LEJOS EV3 afin de programmer le robot. Il est nécessaire d'avoir un bon réseau internet sur lequel nous connectons simultanément le robot et l'ordinateur. Nous avons également créé un Google drive afin de retranscrire nos évolutions et partager nos recherches personnelles.

9.3. Autres contraintes :

Ce projet nécessite la mobilisation de nos connaissances dans le langage JAVA, mais également des recherches sur l'utilisation de la machine virtuelle LEJOS.

Il est aussi important de programmer une réunion par semaine afin de se rendre au Fablab MSTIC pour tester nos programmes sur le terrain de jeu, faire le point sur l'avancée de nos objectifs et mettre en commun nos recherches personnelles.

Enfin, l'utilisation de la plateforme GitHub est nécessaire et obligatoire pour ce projet. Elle permet de stocker et de partager du code. Le code est disponible et modifiable via GitHub, il est possible d'assigner des tâches à des membres en particulier, et les problèmes sont signalés grâce aux issues. Chaque membre du groupe peut créer une issue ou en commenter une pour aider à la résolution de celle-ci. Il est aussi possible de partager les documents du projet via la page wiki.

10. Déroulement du projet

La répartition du travail sur les 12 semaines acquittées va être organisée en fonction des livrables à rendre :

Livrable	Intitulé
1	Cahier des charges
2	Plan de développement
3	Plan de tests
4	Code source
5	Java Documentation
6	Rapport final

Semaine	1	2	3	4	5	6
Date	07/09/2020	14/09/2020	21/09/2020	28/09/2020	05/10/2020	12/10/2020
Intitulé cours	Définition objectifs	Analyse besoin	Spécification	Conception	Développement et test robot	Développement et test robot

Livrable	Intitulé
1	Cahier des charges
2	Plan de développement
3	Plan de tests
4	Code source
5	Java Documentation
6	Rapport final

Semaine	7	8	9	10	11	12
Date	19/10/2020	02/11/2020	09/11/2020	16/11/2020	23/11/2020	30/11/2020
Intitulé cours	Développement et test robot	Développement et test robot	Développement et test robot	Intégration	Recette	Evaluation

Figure 4: Planification du travail

11. Références

- http://www.lejos.org/index.php http://lig-membres.imag.fr/PPerso/membres/pellier/doku.php?id=teaching:ia:project_lego, *Consulté le 18 septembre 2020.*