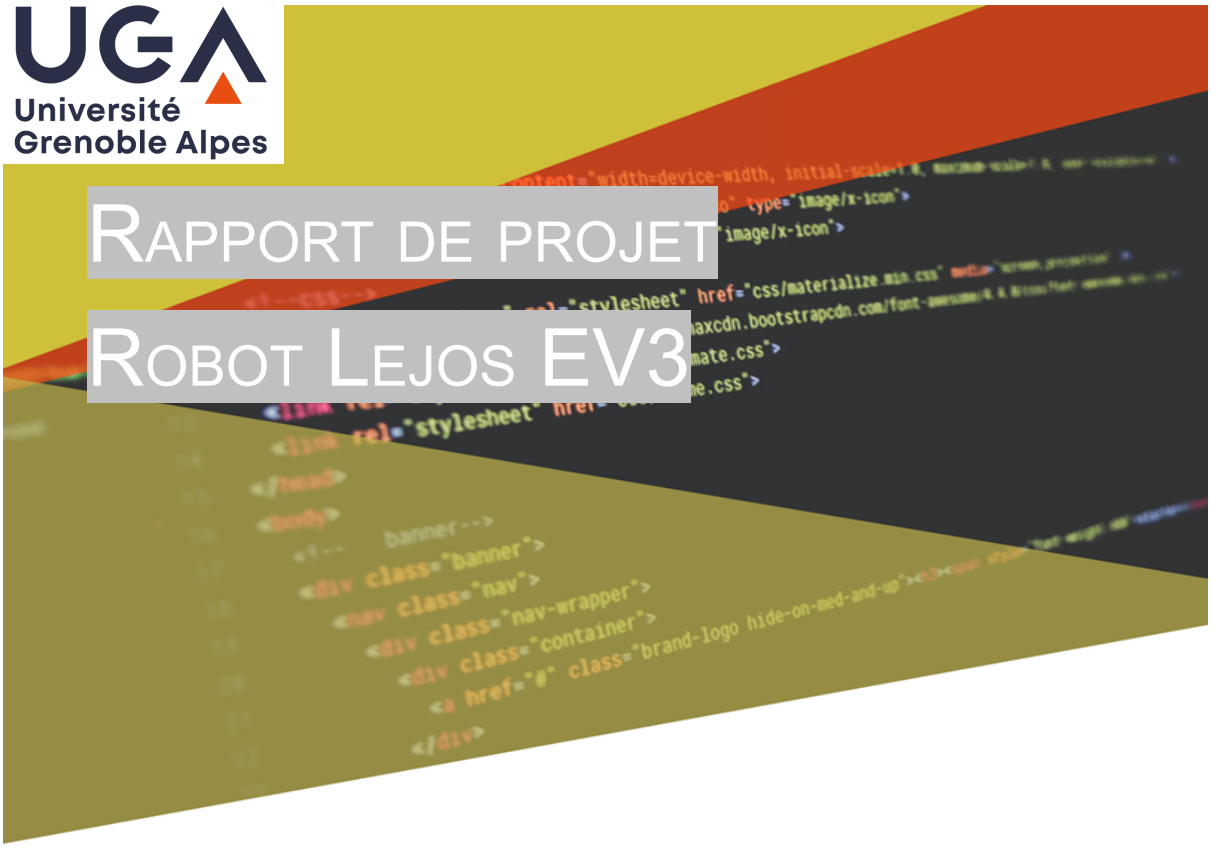
Université Grenoble Alpes



RAPPORT DE PROJET

ROBOT LEJOS EV3

VERSION DU DOCUMENT : 1.0

NOM DU PROJET : ROBOT-LEJOS-EV3-FASTLEARNING

DATE : 03/12/2022

AUTEURS : LOUIS BASTIEN FAUCHON

DANIEL BEQAJ

RAPHAËL DELATTRE

TYPE DE DIFFUSION : HTTPS://GITHUB.COM/VETRARR/-ROBOT-LEJOS-EV3-FASTLEARNING

MAÏTRE D'OUVRAGE : DAMIEN PELLIER **MAÎTRE D'OEUVRE : VOIR AUTEURS**

Introduction

1.1 Objet

Ce rapport marque la finalité du projet que notre groupe de travail a réalisé lors du cours d'Intelligence Artificielle du Semestre 5. Le but du projet était la programmation d'un robot lego, en exploitant la librairie Lejos EV3. L'objectif étant la participation à une compétition avec les autres équipes. Cette compétition consistait à une collecte de palets sur un terrain (2m par 3m) avec des règles bien spécifiées dans la réglementation en annexe du cahier des charges. Ce rapport vise à traduire la réalité de la gestion du projet.

1.2 Documents de référence

Cahier des charges :
https://github.com/Vetrarr/-Robot-Lejos-EV3-FastLearning/blob/main/Cahier%20Des%20Charges_v1.pdf

Plan de développement :
https://github.com/Vetrarr/-Robot-Lejos-EV3-FastLearning/blob/main/Plan%20de%20d%C3%A9veloppement_v1.pdf
Plan de tests :
https://github.com/Vetrarr/-Robot-Lejos-EV3-FastLearning/blob/main/Plan%20de%20test_v1.pdf
Code source et documentation interne

2 Guide de lecture

Ce rapport est le dernier document de la série des documents réalisés pour ce projet. Il vise dans une première partie à décrire l'organisation et la gestion du projet, pour se focaliser ensuite sur les différentes étapes de réalisation et se termine par un bilan comparatif entre les objectifs de départ et les objectifs tenues.

3 Organisation et gestion du projet

Le projet a été réalisé les lundis après midi de septembre à novembre 2022, et à d'autres moments de la semaine selon l'avancement de la programmation avec pour objectif d'avoir des classes fonctionnelles pour le jour de la compétition. Les cahier des charges a permis d'identifier les contraintes et les objectifs du projet, tandis que le plan de développement et le plan de test ont permis d'identifier les étapes d'avancement et de maintenir une gestion appropriée des temps de travail.
La répartition des tâches à été dynamique en fonction de l'avancement et des besoins de la semaine courante, c'est une gestion évolutive et agile qui nous a permis de répartir le travail entre les membres du groupe.
Plusieurs outils sont utilisés pour organiser le projet comme par exemple: discord, google drive, google docs, Eclipse, GitHub etc. Le discord a été utilisé pour la communication courante du projet mais aussi pour le partage de fichier et les appels vocaux. Google drive a été utilisé comme répertoire des documents en préparation avant le dépôt sur GitHub, Google doc a été utilisé pour la rédaction des documents. La programmation a été faite sous Eclipse et n'ayant pas une organisation prédéfinie pour le partage des programmes nous avons abandonné GIT pour un partage des classes les plus récentes.
Nous avons respecté le cahier des charges, le plan de développement et l'échéancier prévisionnel mais certains objectifs n'ont pas été atteints, ceux-ci dépendant de la stratégie. Nous avons identifié les méthodes essentielles au fonctionnement du robot mais plusieurs stratégies étaient possibles. Nous avons privilégié un fonctionnement indéterminé du robot avec néanmoins un choix de différents programmes selon le point de départ sur le terrain.

4 Etapes de réalisation du projet

Etape 1 : Prise en compte des objectifs et du matériel pour la réalisation du projet

Nous avons effectué une vérification du matériel et pris connaissance de toutes les informations mise à disposition, sur le site de D. Pellier, et la documentation du robot disponible sur https://lejos.sourceforge.io/.

Etape 2 : Contrôle du robot avec l'EV3 Control Center sur Eclipse

Après installation du plugin eclipse et connexion au réseau wifi commun entre le pc et le robot, nous avons testés les possibilités offerte par le panneau de control.

Etape 3 : Test des capteurs et valeurs de références

Cette étape a nécessité la prise en main du robot et l'évaluation des capteurs tel que décrite dans le plan de test par mise en situation selon le capteur évalué. De nouvelles mesures ont été effectuées au gré de l'évolution des méthodes au cours du projet.

Etape 4 : Utilisation de la librairie de l'EV3

Nous avons identifié les classes de la librairie qui répondait aux objectifs définis préalablement. Ces objectifs était l'utilisation des actuateurs et senseurs du robot afin de réguler le déplacement du robot selon son environnement, le terrain de la compétition et la détection des palets présents aux intersections des lignes de couleurs.

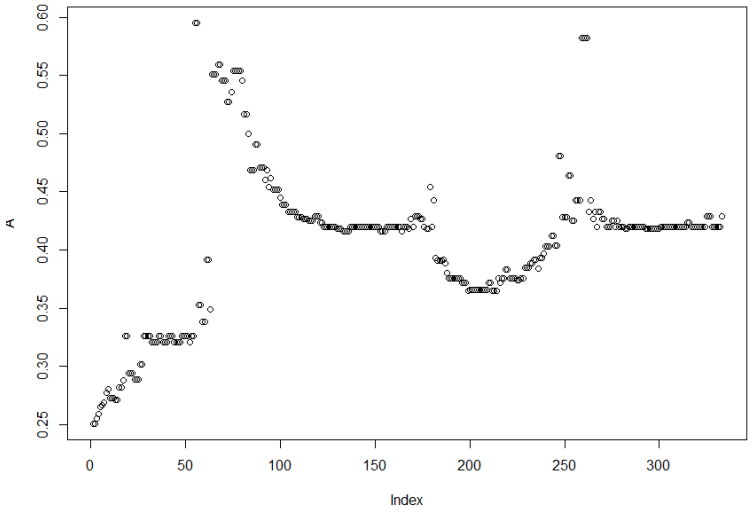
Etape 5 : Définitions des classes

Dans l'objectif de structurer le programme du robot, différentes fonctions ont été classées selon leurs caractéristiques communes. Chaque grande fonction du robot a été définie dans des classes java sur eclipse. Les fonctions de détections dans la classe Senseurs, les fonctions de déplacement dans la classe Moteurs, les fonctions d'interaction entre les senseurs et les actuateurs dans la classe Réflexion, et les fonctions d'états du robot dans la classe Robi.

Etape 6 : Définitions des méthodes de la programmation du robot

Les méthodes utilisent la librairie à disposition et correspondent à des fonctions plus ou moins complexes du robot. Des méthodes de déplacement ou une méthode de détection particulièrement difficile à définir. Une première étape de définition de la méthode s'est basée sur les informations données par la documentation et une suggestion du maître

d’ouvrage. Finalement le graph tracé pendant la rotation du robot en présence d’un palet a proximité montre un pattern permettant la détection.

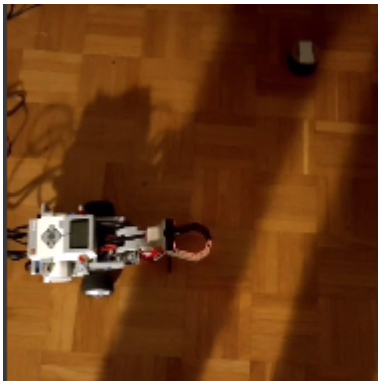


Etape 7 : Définitions des états du robot

Les états du robot correspondent aux états depuis le début d’une manche de la compétition jusqu’à son terme. Chaque état étant déclenché selon un scénario prédéfini, la rencontre avec un palet, le robot adverse qui coupe la trajectoire, un but …

Etape 8 : Tests du robot

Les tests ont été réalisés aux différentes étapes de développement, les tests les plus élémentaires étaient au niveau des capteurs et actuateurs, puis les différentes fonctions, les états n’ont pas pu être testé autant que souhaité ceux-ci nécessitant une plus grande variété de tests.



Etape 9 : Compétition

Pour la compétition le robot a un programme embarqué et est capable de saisir des palets pour les déposer dans l'en-but adverse. Le choix du programme se fait sur l’écran du robot. La phase des qualifications a été atteinte, s’est succédé les phases de pools qui n’ont pas été dépassées.

Explication du code du robot

La classe Senseurs

Contient :

- Un constructeur senseurs() qui a pour but d'initialiser les attributs ultraSensor() ,touchSensor(), et colorSensor().
- Une méthode getDistance() qui retourne la distance entre la caméra infrarouge du robot et l'obstacle le plus proche
- Une méthode isTouch() qui retourne un boolean qui vaudra true si le capteur est appuyé
- Une méthode getColor() qui renvoit un int en fonction de la couleur capté par le capteur situé sous le robot

La classe Moteurs

Contient :

- Un constructeur Moteurs() qui a pour but d'initialiser l'attribut pilot en commençant par les moteurs des roues gauche et droite, qui vont servir à initialiser un attribut chassis qui sert enfin à initialiser l'attribut pilot qui sera utilisé pour toutes les méthodes suivantes.
- Une méthode forward(double distance) qui permet de faire avancer le robot d'un distance donnée "distance"
- Une variante de forward qui s'appelle forwardAsync(double distance) prenant également une distance donnée "distance" mais cette fois ci ne bloquant pas le code à cette étape permettant de continuer sur la suite du code sans avoir fini le forward.
- Une méthode stop() permettant d'arrêter tout les moteurs d'un coup
- Une méthode rotate(float angle) qui fait faire au robot une rotation de angle degré et sa variante rotateAsync(float angle).
- Une méthode isMoving() qui renvoie un boolean qui vaut true si le robot est en mouvement false sinon.
- Une méthode ouvrirPinces() et une fermerPinces() qui permettent d'ouvrir et fermer les pinces.

La classe Reflexion

Contient :

- Un constructeur Reflexion qui initialise les attributs moteurs et senseurs qui permettront d'appeller des méthodes des classes Senseurs et Moteurs dans cette classe.
- Une méthode attraperPaletDevant() qui appelle la méthode attraperPaletDevant(float distance) avec pour paramètre le float retourné par la méthode getDIstance de la classe Senseurs.

- La méthode attraperPaletDevant(float distance) qui ouvre les pinces,fait avancer le robot puis ferme les pinces si la méthode isTouch renvoie true.
- La méthode chercherPalet(int degreRotation) qui set la vitesse à 10 degré par secondes, créé une arrayList de tableaux de float echantillon dans laquelle vont etre entré les valeurs de getDistance et leur indice dans le tableau puis on regarde chaque valeur des distances pour trouver la plus petite et retourner l'angle à laquelle cette valeur a été trouvé par un produit en croix entre le nombre de données prises, l'angle total de la rotation et l'indice de la distance la plus proche.
- Une méthode allerBut() qui fait faire une rotation pour se tourner face au but en utilisant l'attribut angleAuBut de la classe Moteurs qui est actualisé à chaque rotation effectué par le robot. Ensuite le robot avancera tout droit jusqu'a ce que le capteur couleur ne trouve une correspondance avec la ligne couleur blanche (valeur 6 ici).

La classe Robi

Contient :

- Un début de programme qui fait patienter le robot qui demandera si il commence le match à gauche ou à droite ce qui permettre d'initialiser l'attribut etatCourant vers ETA1G ou ETA1D
- Un switch avec plusieurs états :
  -ETA1G/ETA1D qui est une suite de commandes simple et non aléatoire permettant au robot d'aller chercher le palet devant lui et de l'amener le plus vite possible au but adverse tout en esquivant le robot adverse si besoin

  -ETA2 qui est la recherche de palet, le robot se mettre parallèle à un mur de façon à ne pas le scanner et tournera de 90 ou -90 degré en fonction de la ou il a commencé, ensuite si il a trouvé un palet devant lui il va directement à ETA3 sinon il passera à ETA4 afin de corriger son angle pour se tourner face au palet et aller à ETA3

  -ETA 3 qui avance tout droit jusqu'à attrapet le palet et l'amener au but

  -ETA 4 qui tourne le robot face au palet

-Et enfin en dehors du switch une méthode secuAsync qui est appelé à chaque déplacement du robot qui est un arrêt de secours pour que le robot ne rentre pas dans un mur ou un robot adverse, qu'il tourne à 90 degré et en fonction de si le robot tient un palet ou non change l'état courant à ETA3 ou ETA2.

Bilan du projet

Lors de la compétition, notre groupe (Robi) est attribué au Groupe A de la compétition,donc celle des 4 groupes ayant marqué le plus de palets lors de la phase des qualifications .On n'a pas réussi à passer la phase des groupes, mais on est classé les troisième du "meilleur" groupe.
Le robot a été capable de saisir les palets et les déposer dans l'en-but adverse, mais aussi de réagir dans différentes situations d'interruption. Le robot a également été capable de saisir des palets se trouvant hors des intersections des lignes de couleurs sans utiliser la caméra IR. Des objectifs supplémentaires de parcours prédéterminé auraient pu être mis en place par l'utilisation de l'ensemble des capteurs en optimisant les trajectoires selon les déplacements récurrents des palets. Néanmoins il aurait fallu simuler un plus grand nombre de manches et réaliser ces simulations en présence d'un robot adverse.

Glossaire

Définit l'ensemble des termes spécialisés du document
Intelligence Artificielle = un agent intelligent qui perçoit son environnement et agit de façon pour maximiser les chances d'accomplir ses objectifs.Ici: le robot pourra reconnaître son environnement grâce à différents capteurs et faire des actions dit "intelligentes" comme par exemple éviter la collision avec les autres robots.
Robot autonome = un robot autosuffisant qui n'exige pas d'assistance humaine.Ici:un robot qui exécute son programme et agit pour accomplir ses objectifs sans besoin d'être contrôlé à distance.
Robot perdu = un robot qui ne répond plus

Référence

-Le règlement de la compétition
https://lig-membres.imag.fr/PPerso/membres/pellier/doku.php?id=teaching:ia:project_lego
-Lejos
https://sourceforge.net/p/lejos/wiki/Home/

Index

Annexe