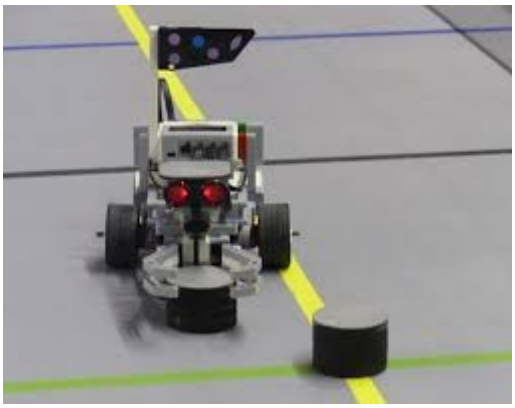
Projet de robotique
-
Lego Mindstorm



Cahier des charges

Membres :
-Balatti Enzo
-Bellatif Mamoune
-Bergman Clément

L3 MIASHS
Année 2019-2020

Table des matières

Introduction

Contexte

Ce projet s'inscrit dans le cadre d'un cours d'initiation à l'intelligence artificielle, qui s'étend sur un semestre. Le but de ce projet est de créer un programme permettant à des robots MindStorm de participer à des matchs. Lors des matchs, deux robots s'affrontent, le but est de saisir le plus de palets possibles disposés sur un terrain, et de les déposer dans le camp adverse. Le robot qui aura ramené le plus de palets gagne la partie. A la fin du semestre le robot doit être capable de jouer.

Description de la demande

Les objectifs

L'objectif du projet est de programmer un code permettant à un robot Lego Mindstorm de jouer selon les règles et de battre les autres robots.

Il doit être capable de repérer les palets sur le terrain, de les saisir un par un et de le déposer dans la zone d'en-but adverse, puis de revenir sur le terrain et de repérer d'autres palets. L'objectif principal est d'obtenir le maximum de points à l'issue du temps réglementaire.

Produit du projet

Ce qui doit être produit a l'issue de ce projet est un programme, qui lorsqu'on l'exécute, permet au robot de jouer. Ce programme est constitué d'une classe principale appelée Robot. Cette classe contient plusieurs méthodes et une méthode *main* qui permet d'exécuter le programme.

La stratégie de notre programme est simple. Premièrement il faut aller marquer en premier, car le premier robot qui marque lors d'un match reçoit un bonus de 2 points (quand chaque autre palet marqué rapporte 3 points, un palet dans les pinces à la fin du match rapporte 2 points). Une fois cette étape passée, le robot doit effectuer des rotations sur lui même afin de détecter les palets les plus proches. Une fois le palet le plus proche détecté, le robot doit aller marquer et recommencer l'opération durant le reste du match, ou jusqu'à ce qu'il n'y ait plus de palets sur le terrain.

Pour pouvoir se repérer dans le terrain et savoir où aller pour marquer, le robot enregistre en mémoire son inclinaison (appelé angle courant) par rapport à sa position de départ (gauche, milieu ou droite, voir la section contraintes spatiales). Si son angle courant est nul, le robot est placé en face de l'en-but. De plus le côté de son en-but est enregistré grâce à la fonction *avantMatch* pour pouvoir aller dans le bon en-but une fois un palet attrapé, et pour pouvoir se repositionner si celui-ci est désorienté pour n'importe-quelle raison.

Si le robot se retrouve désorienté, il doit être capable de se réorienter. Pour cela il lance une procédure de replacement afin de reprendre le jeu (la procédure est expliquée dans le détail des fonctions plus bas).

Les fonctions du produit

Fonctions pour la détection des couleurs :

-*calibrageCol* : Cette méthode est appelée une seule fois avant les matchs, pour stocker dans un fichier texte les valeurs correspondant aux couleurs présentes sur le terrain. Ce fichier sera utile pour détecter les couleurs. Les couleurs enregistrées sont le blanc, le noir, le jaune, le rouge, le bleu et le vert.

-*getCouleur* : Cette méthode permet de déterminer la couleur captée par le capteur de couleur, en comparant ses composantes rouges, vertes et bleues (RVB) à celles des couleurs stockées dans le fichier texte. La couleur détectée est celle dont les composantes RVB sont les plus proches parmi celles stockées au sein du fichier texte. Elle renvoie une chaîne de caractères correspondant à cette couleur.

Fonction liée à la détection de prise de palet :

-*paletDansPinces* : Cette méthode doit permettre de savoir si un palet est entre les pinces. Pour cela, un capteur de pression est positionné au dessus des pinces et s'active lorsqu'un palet enfonce une tige reliée à ce capteur.

Fonctions diverses :

Ces fonctions sont des fonctions de calcul, afin de permettre aux fonctions de détection (capteurs de couleur et de distance via ultrason) d'être fonctionnelles

-*scalaire* : Cette méthode permet de calculer le scalaire entre 2 paires de 3 valeurs . En faisant la racine carrée des sommes des différences entre chaque paire, on obtient l'écart total entre ces 2 paires. C'est idéal pour calculer la proximité entre 2 couleurs, qui ont chacune 3 composantes RVB.

-*getMin*: Cette méthode permet de retourner la valeur minimale (dans les limites de détection d'un palet) d'une liste. Par exemple, si une valeur inférieure à la capacité de détection minimale d'un palet est présente dans cette liste, elle ne sera pas prise en compte.

Fonctions de déplacement

-*avancerDe* : Cette fonction permet au robot d'avancer selon une certaine distance en millimètres, donnée en paramètre. Une distance négative permet de reculer.

-*virage* : Cette fonction permet au robot d'effectuer une rotation, en tournant ses deux roues simultanément, selon un angle en degré donné en paramètre. Si l'angle donné est négatif le robot tournera vers la gauche et inversement vers la droite. Il est important de ne pas effectuer d'autre déplacements pendant l'appel de cette méthode, afin de ne pas dérégler la position angulaire (angle courant) du robot et éviter que orientation vers l'en-but soit faussée.

Fonctions de jeu :

Ces fonctions sont liées au jeu en lui-même. Celles-ci s'appellent entre elles par moment et utilisent certaines fonctions détaillées précédemment (déplacement, utilisation des capteurs) ou intègrent de nouvelles fonctionnalités (détection de distance via capteur ultrason).

-*avantMatch* : Cette méthode est appelée avant le début d'un match, elle permet d'indiquer au robot où il va commencer afin de garder en mémoire la position de l'en-but et de pouvoir marquer le premier palet.

-*marquerPremierPalet* : Cette fonction, lancée dès le début du match s'occupe comme son nom l'indique de marquer le premier but. Sa durée d'exécution doit être minimale pour pouvoir obtenir les points bonus attribués au premier buteur.

-*chercherObstaclePlusProche*: Effectue un balayage de l'environnement du robot tout autour de lui pour ensuite aller chercher le palet le plus proche et enfin marquer. Cette fonction est importante car il est possible que le robot se trompe et détecte un objet n'étant pas un palet.

Il faut s'assurer que l'objet le plus proche n'est pas un robot ni un mur. Il faut donc avancer jusqu'à cet objet, et s'arrêter à proximité de lui de manière à ce que le robot ne détecte plus l'objet si c'est un palet. Si la distance que l'on obtient à ce point est supérieure à la distance attendue, alors c'est un palet. Cela peut aussi être un robot, si ce dernier a bougé

entre les deux mesures de distance. Sinon c'est un mur. Si le robot détecte que c'est un palet ou un robot, il avance jusqu'à l'objet identifié et ouvre ses pinces, si le capteur de pression est activé alors c'est un palet, sinon l'objet détecté était un robot qui a bougé depuis.

Il faut éviter d'aller en direction des murs, et d'aller chercher des objets situés dans les en-buts. La zone de recherche est donc limitée à la zone comprise à l'intérieur des deux lignes blanches, de la ligne rouge et de la ligne jaune évitant un accès vers les murs, et les en-buts. Théoriquement, aucun palet encore en jeu ne se trouve en dehors de cette zone : ils sont placés aux intersections des lignes de couleurs. Une fois un but marqué, cette fonction se replace dans la zone de recherche et recommence un balayage. Elle est donc récursive.

Voir en annexe la zone de recherche dessinée sur une représentation du terrain.

-*ramasserPalet* : Cette fonction est utilisée lorsqu'un palet est détecté. Le robot attrape le palet en question à l'aide de ses pinces. Si une ligne blanche est détectée ou qu'aucun palet n'est ramassé on relance la recherche d'un palet.

-*marquerPalet* : Lorsque cette fonction est appelée, le robot s'oriente vers la zone d'en-but adverse grâce à son orientation stockée en mémoire et avance jusqu'à celle ci pour y déposer le palet.

-*deposerPalet* : Lorsque cette méthode est appelée, le robot ouvre ses pinces, recule, tourne de 180 degrés et ferme ses pinces.

<u>Fonctions liées au replacement</u>:

Quand le robot est désorienté, il peut se replacer dans l'axe du but en lançant une procédure de replacement.

-*faceAuMur* : Le robot effectue un balayage autour d'un mur pour se placer à la perpendiculaire par rapport à lui. A partir de là il est prêt à se replacer

-*replacement* : Une fois le robot placé face au mur, il effectue un demi-tour et avance jusqu'à identifier la couleur des deux premières lignes de couleurs qu'il croise.Il lance la méthode *comparaison* afin de se repositionner.

-*comparaison* : Cette fonction est liée à la précédente car c'est elle qui permet d'étudier la combinaison des deux couleurs enregistrées de la position de l'en-but afin d'effectuer les rotations nécessaires pour être replacé en direction du but avec un angle courant mis à jour. Voir en annexe les différentes combinaisons possibles.

-*procedureDeReplacement* : Cette fonction lance toute la procédure de replacement. Elle fait donc appel aux 3 précédentes fonctions dans l'ordre.

Contraintes

Contraintes de délais

Plusieurs rendus sont attendus. Le 23 septembre, la première version du cahier des charges est à rendre. Le 7 octobre, le plan de développement est à rendre, qui détaillera comment et dans quel ordre sera développé le programme. Le 18 novembre, le plan de tests sera à rendre, ce document détaillera comment les tests ont été réalisés tout au long du développement. Il faudra également rendre le code source et la documentation interne du code ce jour la. Le rapport final sera a rendre le 13 décembre.

Contraintes matérielles

Le matériel utilisé est un robot Lego Mindstorm EV3. Sa construction est établie au préalable. Il est composé d'un boîtier central où sont stockés les programmes.

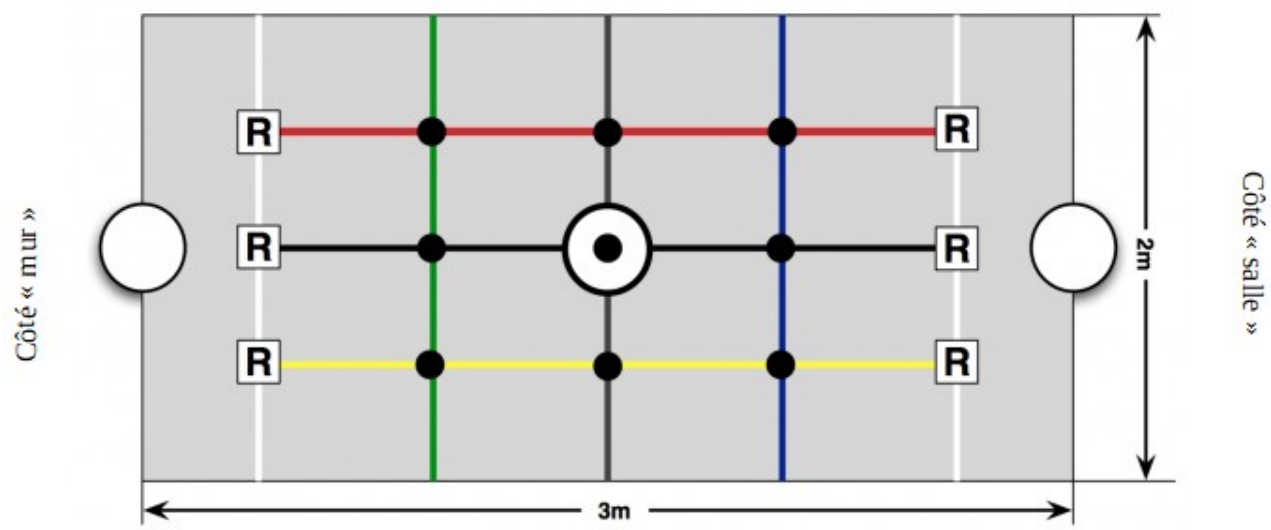
Grâce à ses moteurs, le robot est capable de faire tourner ses roues, indépendamment l'une de l'autre ou simultanément, et donc de se déplacer dans toutes les directions. Cela lui permet d'avancer et de reculer, plus ou moins vite, de pivoter, c'est à dire qu'il tourne sur lui même, dans ce cas une roue avance et l'autre recule. Il doit être capable de le faire dans les deux sens (en tournant à sa droite ou à sa gauche), et il doit pouvoir pivoter autant que nécessaire. Il doit être capable de s'orienter vers un palet et le saisir.
Le robot peut attraper le palet grâce à sa pince située à l'avant.
Si le détecteur de pression placé à l'avant est actionné on souhaite que le robot attrape le palet en ouvrant et en refermant ses pinces. A l'inverse lorsque le robot sera dans la zone d'en-but adverse, il ré-ouvrira les pinces pour déposer le palet.
Les robots seront sur un terrain de 3*2m fermé par des bordures rigides transparentes de 15cm de hauteur. Le terrain est divisé en 3 parties : les 2 zones d'en-but, ainsi que le terrain principal où 3 bandes de couleurs horizontales et verticales divisent le terrain. Chacun des 9 palets est positionné à l'intersection de chaque bande.

Le robot Lego Mindstorm EV3 est relié aux 3 moteurs et aux 3 capteurs du robot. Il dispose d'un moteur pour chaque roue ainsi qu'un autre pour actionner les pinces qui attrapent les palets. Il dispose d'un capteur de pression; d'un capteur de son et d'ultrasons placé à l'avant du robot mais aussi un capteur de lumière placé lui en dessous du robot.
Ce robot dispose d'une batterie dans son boîtier central. Celle-ci peut être chargée par une prise secteur. Une carte SD est insérée dans le boîtier central qui permet d'interagir avec des programmes java qui sont eux envoyés par un réseau wifi local depuis un ordinateur et reçus sur le boîtier grâce à une clé wifi branchée dessus.
Des Programmes codés sur le langage java sont envoyés via wifi depuis un ordinateur portable sur une clé wifi connectée au robot.
Il faut que la luminosité soit suffisante pour que le robot puisse suivre les lignes colorées tracées sur le terrain.
La brique programmable du robot a les caractéristiques suivantes:
•Microprocesseur 32 bit ARM7 d'ATMEL,

•Fonction Bluetooth (connexion à d'autres NXT ou à un PC et possibilité de contrôler le NXT avec un téléphone portable ou un autre appareil Bluetooth).

Contraintes spatiales

Le terrain est de couleur grise (couleur non enregistrée dans notre fichier de couleur) de 3 mètres par 2 (largeur des en-buts). Chaque palet est placé à chaque intersection entre 2 lignes de couleurs (9 au total). La position de départ du robot est indiquée par le R sur le schéma.



Autres contraintes

Aucun plagiat n’est accepté. Le robot doit être conforme aux plans fournis pour la construction du robot.

Déroulement du projet

Planification

Il faut tout d’abord établir un échéancier ainsi qu’un cahier des charges. La seconde phase du projet sera de créer les automates pour représenter le fonctionnement ainsi que le lien des différentes fonctions entre-elles. Avant d’entamer, la partie de développement du programme, il sera nécessaire de créer le diagramme de classes (UML) puis lancer le développement: c’est à dire de coder les classes, méthodes du programme et de les tester au fur et à mesure. Cette partie est celle qui prendra le plus de temps lors du déroulement de ce projet. La rédaction du dossier final et du rendu du code sera la toute dernière étape.

Ressources

On dispose d’un robot lego Mindstorm EV3, d’un palet, et d’ordinateurs portables. Le groupe est composé de trois membres.

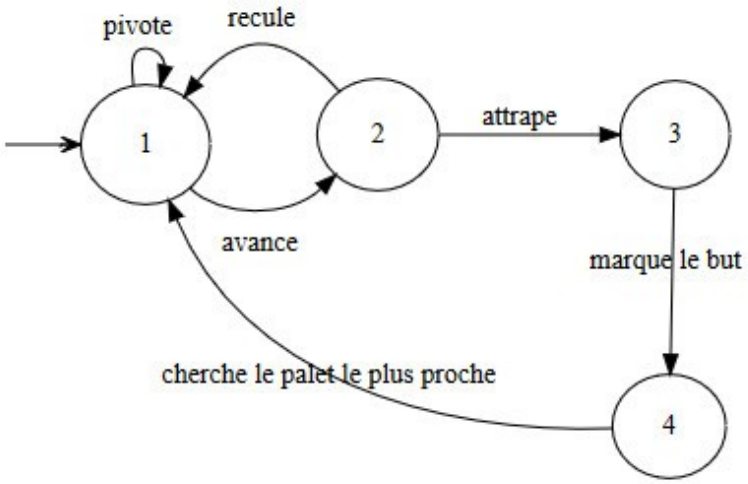
Organisation

Les tâches sont réparties entre les membres de l’équipe, aucun chef de projet n’est désigné. Les mises en commun se feront lors du temps prévu pendant des séances de cours ou lors de réunions de groupes organisées hors des séances de cours.

Annexes

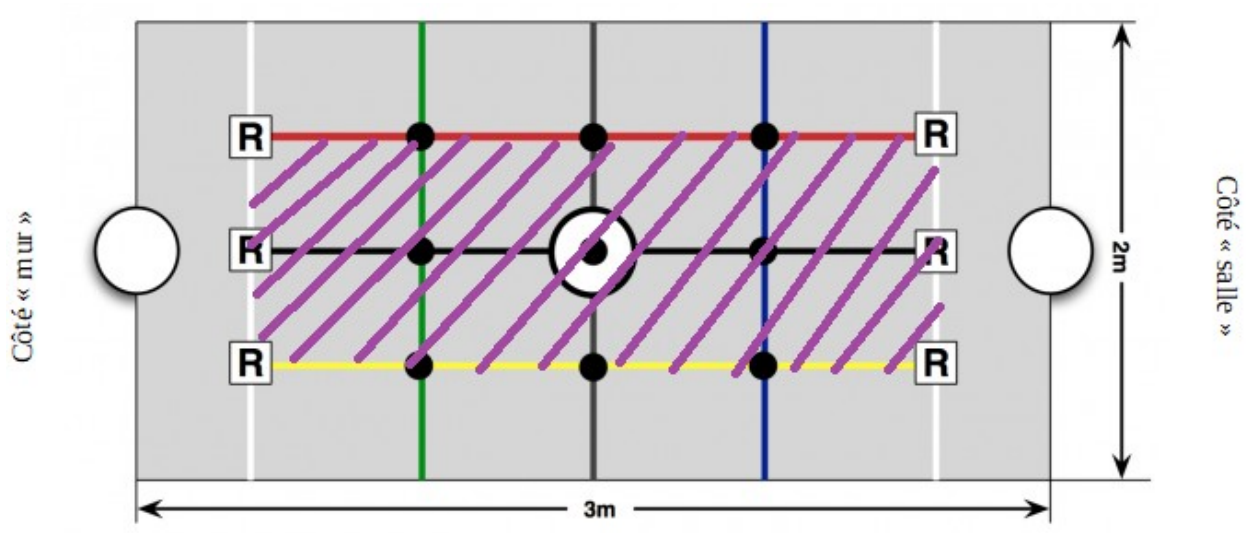
Automate de départ

voici l'automate représentant le fonctionnement le système du robot. Ce diagramme est susceptible d'être modifié durant la phase de développement.



Zone de recherche

La zone de recherche de palets pour notre robot est la zone hachurée en violet ci-dessous. La création d'une zone de recherche limitée par rapport au terrain entier est justifiée par la volonté d'éviter au robot d'aller en direction des murs latéraux et des en-buts. Peu importe l'objet situé dans un en-but (palet, robot, autre), il n'est d'aucun intérêt pour le robot de s'y intéresser. Éviter les contacts avec les murs permet de prévenir tout risques (hors contact avec un robot) de désorientation du robot. Notre stratégie est donc d'éviter de se perdre, de se désorienter par rapport à l'en-but quitte à diminuer nos chances de ramasser des palets (dans les cas où des palets se retrouveraient hors de la zone de recherche)



Combinaisons possibles pour le replacement

Lors de la procédure de replacement, une fois le robot aligné avec les lignes blanches grâce à la fonction *faceAuMur* et en direction du terrain, il est possible de se replacer en fonction des couleurs des deux premières lignes traversées et de la position de notre en-but (côté mur ou côté salle). Si la première couleur détectée est blanche, on fait demi tour et on détecte la suivante Voici les différentes issues possible et les degrés de rotations nécessaires au replacement :

Si l'en-but est côté **salle** le robot est

-aligné (rotation de 0°) si : Vert + Noir OU Noir + Bleu OU Blanc + Vert Ou Bleu + Blanc

-en sens inverse (rotation de 180°) si : Noir + Vert OU Bleu + Noir OU Vert + Blanc OU Blanc + Bleu

-orienté à gauche(rotation de 90°) si : Jaune + Noir OU Noir + Rouge

-orienté à droite (rotation de -90°) si : Noir + Jaune OU Rouge + Noir

Si l'en-but est côté **mur** le robot est

-aligné (rotation de 0°) si : Noir + Vert OU Bleu + Noir OU Vert + Blanc OU Blanc + Bleu

-en sens inverse (rotation de 180°) si : Vert + Noir OU Noir + Bleu OU Blanc + Vert Ou Bleu + Blanc

-orienté à gauche(rotation de 90°) si : Noir + Jaune OU Rouge + Noir

-orienté à droite (rotation de -90°) si : Jaune + Noir OU Noir + Rouge