Plan de Développement

1. Introduction Générale

1.1 Contexte

Le projet d'intelligence artificielle porte sur la programmation d'un robot Mindstorms EV3 afin qu'il puisse détecter, manipuler et transporter des palets de manière autonome. La programmation sera effectuée en Java via l'IDE Eclipse, en utilisant la librairie LeJOS.
En fin de semestre, les divers groupes s'affrontent lors de plusieurs matchs. L'équipe qui parviendra à déplacer le maximum de palets dans le temps imparti sera désignée gagnante.

1.2 Objectif du Plan

Ce plan a pour objet de spécifier les différentes étapes du développement de ce projet en définissant clairement les tâches à accomplir et leur répartition dans le temps. De plus, il offre aussi aux différents membres du groupe ainsi qu'au commanditaire une vision globale de l'avancement du projet, facilitant ainsi la coordination et le suivi du travail.

2. Analyse Préliminaire et Ressources

2.1 Ressources Matérielles

Nous disposons d'un robot EV3 LEGO Mindstorms sur lequel nous implémentons un programme permettant d'interagir avec le robot de manière asynchrone. Nous avons également accès à un terrain pour tester notre robot.
Sélection des capteurs EV3 utilisés dans le programme :

- Moteurs / Capteurs :
  - Roues :
    - Droite (Port M.A) : responsable de la rotation de la roue droite.
    - Gauche (Port M.B) : responsable de la rotation de la roue gauche.
  - Pinces (Port M.D) : utilisé pour ouvrir et fermer les pinces du robot.
  - Ultrason (Port S.4) : utilisé pour mesurer la distance entre le robot et les objets environnants.

- Toucher (Port S.3) : utilisé pour détecter les pressions ou les contacts physiques.
- Détecteur de couleur (Port S.2) : utilisé pour identifier différentes couleurs et intensités lumineuses.

2.2 Contraintes Techniques

Les composants du robot comportent certaines limites qu'il faut prendre en compte lors de la réalisation :

Le capteur d'ultrasons renvoie la distance séparant le robot et l'objet le plus proche, toutefois il est ne précise si cet objet est un palet, un autre robot ou un mur.
De plus, le palet étant petit, il ne peut le détecter que lorsque celui-ci se trouve à une distance comprise entre 32,6 et 107 cm et dans son champ division qui correspond à environ un angle d'environ 30°.

3. Objectifs Techniques du Projet

3.1 Missions Principales :

On cherche à concevoir un robot capable de collecter et déposer un maximum de palets dans un temps imparti, tout en respectant les contraintes du plateau et les règles spécifiques de la compétition. Ainsi, le programme doit permettre aux robots de :

- Localiser les palets à l'aide des capteurs.
- Naviguer sur le terrain de jeu tout en évitant les obstacles (robot adverse, murs)
- Se déplacer vers le camp adverse jusqu'à détection de la ligne blanche
- Déposer le max des palets avant la fin de la manche

3.2 Résultats Attendus

- Vitesse et efficacité : Parmi, les résultats souhaités est que le robot effectue ces actions en un minimum de temps, minimisant ainsi les erreurs telles que les collisions, les rondelles ou les trajectoires sous-optimales. Les stratégies d'optimisation du code et de navigation contribueront à accélérer ces processus tout en offrant des garanties maximales.

- Performances du Robot : Le robot doit être capable de ramasser un maximum de palets dans les meilleurs délais, et puis les déposer après avoir franchi la ligne blanche.

- La conformité aux règles : Le robot doit respecter toutes les règles de la compétition, notamment franchir la ligne blanche avant de déposer les palets dans la zone de dépôt.

4. Architecture du Système : Perception et Action

Package 1 : Perception

Ce package permet la communication avec tous les capteurs qui permettent au robot de percevoir son environnement.

- CapteurCouleur: Permet au robot de détecter les couleurs de lignes sur le terrain.

    - Méthodes :
        - getColorID() : int : retourne l'identifiant numérique de la couleur détectée.
        - `getColorName()` : String : retourne le nom de la couleur détectée en utilisant l'identifiant.
        - isColor(String colorName) : boolean : vérifie si la couleur détectée correspond à une couleur donnée.
        - isWhite() : boolean : retourne true si la couleur détectée est blanche.
        - isBlack() : boolean : retourne true si la couleur détectée est noire.
        - isBlue() : boolean : retourne true si la couleur détectée est bleue.
        - close ( ) : void : éteint le capteur.

- CapteurUltrason: Utilisé pour détecter les objets différents de l'environnement et estimer les distances.

    - Méthodes :
        - getDistance():float : Renvoie la distance mesurée par le capteur.
        - detectPalet():boolean : Renvoie vrai si un palet est détecté.
        - detectObjet(float detectionMin):boolean :retourne true si la distance de l'objet détecté est inférieure à la distance minimale spécifiée.
        - close ( ) :void : éteint le capteur.

- CapteurTouche: Utilisé pour détecter si le robot est en contact avec un objet.

- Méthodes :
    - `isPressed():boolean` : Indique si le capteur tactile a été activé.
    - close ( ) : void : éteint le capteur.

Package 2 : Action

Ce package contrôle les actions physiques du robot.

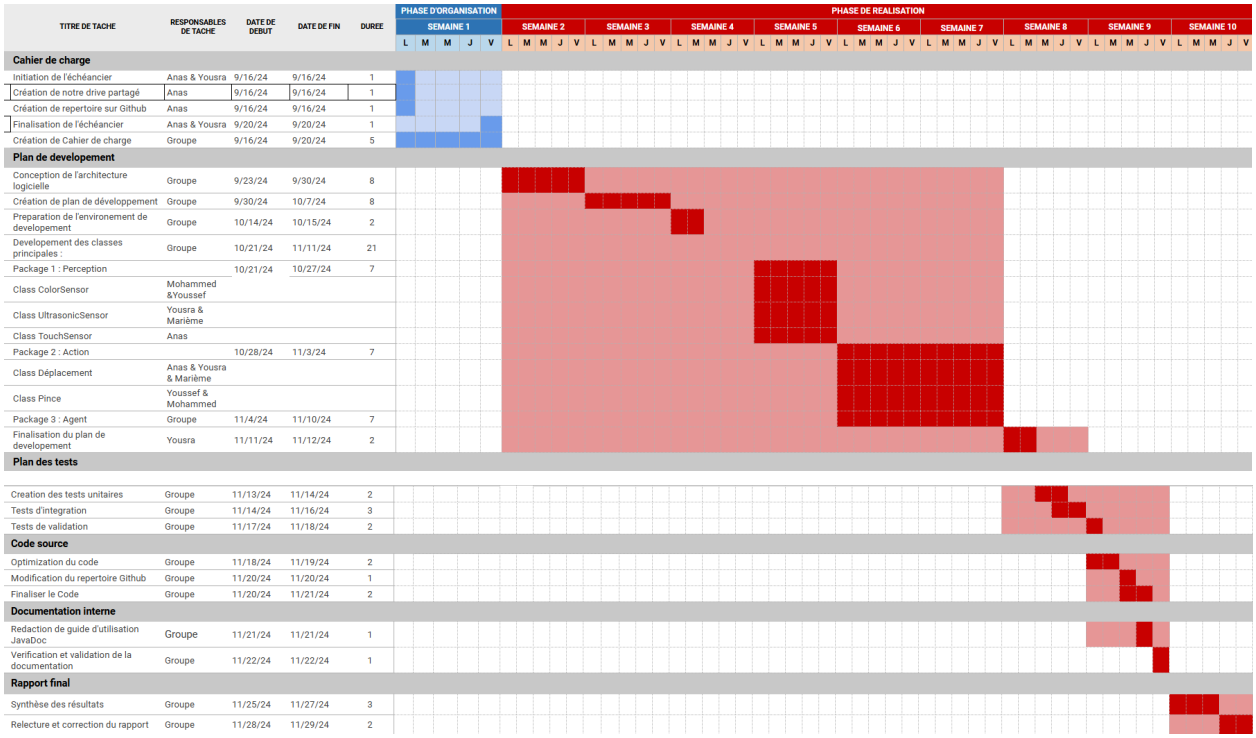
- Déplacement :
    - Méthodes :
        - getDirection():Float: retourne la direction actuelle du robot en degrés.
        - getPilot():MovePilot: retourne l'objet MovePilot.
        - avancerSync(double distance) : void : avance le robot de façon synchrone sur la distance en paramètres.
        - avancer(double distance) : void : avance le robot de façon asynchrone sur la distance en paramètres.
        - tournerSync(double angleToTurn) : void : effectue une rotation synchrone de l'angle en paramètre.
        - tourner(double angleToTurn) : void : effectue une rotation asynchrone de l'angle en paramètre.
        - tourner(double angleToTurn, boolean heading) : void : effectue une rotation avec correction en fonction du PoseProvider si nécessaire.
        - modifVitLin(double s):void :modifie la vitesse de déplacement linéaire en cm par seconde.
        - modifVitRot(double s):void :modifie la vitesse de rotation en degrés par seconde.
        - isMoving ( ) : boolean : retourne true si le robot est en mouvement.
        - stop():void : Arrête immédiatement le mouvement du robot.
- Pince :
    - Méthodes :
        - fermerPince():void : ferme les pinces pour saisir un palet si elles sont ouvertes.
        - ouvrirPince ():void : ouvre les pinces pour relâcher un palet si elles sont fermées.
        - setPincesOuvertes(boolean pincesOuvertes):void : Modifie l'état des pinces.

Package 2 : Agent

- Agent :
    - Méthodes :
        - detecterLesObjets() : ArrayList<float[]> : détecte les objets en effectuant une rotation de 360° et retourne une liste contenant les distances et directions des objets détectés.
        - bestObjet (ArrayList<float[]> objets) : float[] : retourne les coordonnées (distance, direction) de l'objet le plus proche parmi ceux détectés.
        - attraperPalet(boolean premier) : boolean: tente d'attraper un palet, retourne true si l'action a réussi, sinon false.
        - retourStrategie() : void : recule et exécute une stratégie pour gérer d'autres palets.
        - lacherPalet ( ) : void : dépose le palet au camp adverse et se repositionne.
        - devierDeLaLigne() : void : dévie le robot de la ligne de départ pour éviter les obstacles lors de premierPalet.
        - premierPalet( ) : void : effectue l'action pour attraper et marquer le premier palet.
        - deuxiemeEtTroisiemePalet(String direction) : boolean: gère les actions pour attraper et marquer le deuxième ou troisième palet, selon la direction donnée (Gauche ou Droite).
        - Surveiller( ) : boolean : surveille les obstacles pendant que le robot se déplace, retourne true si un obstacle est détecté et évité, sinon false.
        - marquerPalet( ) : void : oriente le robot vers le camp adverse et marque un palet.
        - versCouleur(String couleurCible) : void : déplace le robot jusqu'à détecter une couleur spécifique.
        - esquive() : void : permet d'éviter des obstacles en adaptant la trajectoire du robot.
        - chercherPalet() : boolean : détecte et se déplace vers le palet le plus proche, retourne true si un palet est trouvé, sinon false.
        - ligneCentrale() : void : déplace le robot d'une distance de 120 cm afin d'atteindre la ligne centrale qui lui sert d'indicateur, tout en surveillant les obstacles.
        - autrePalets() : void : implémente une stratégie pour chercher et gérer les palets restants.

- run() : void : programme principale

5. Planification du Projet



TITRE DE TACHE	RESPONSABLES DE TACHE	DATE DE DEBUT	DATE DE FIN	DUREE
Cahier de charge				
Initiation de l'échéancier	Anas & Yousra	9/16/24	9/16/24	1
Création de notre drive partagé	Anas	9/16/24	9/16/24	1
Création de repertoire sur Github	Anas	9/16/24	9/16/24	1
Finalisation de l'échéancier	Anas & Yousra	9/20/24	9/20/24	1
Création de Cahier de charge	Groupe	9/16/24	9/20/24	5
Plan de developement				
Conception de l'architecture logicielle	Groupe	9/23/24	9/30/24	8
Création de plan de développement	Groupe	9/30/24	10/7/24	8
Preparation de l'environement de developement	Groupe	10/14/24	10/15/24	2
Developement des classes principales :	Groupe	10/21/24	11/11/24	21
Package 1 : Perception		10/21/24	10/27/24	7
Class ColorSensor	Mohammed &Youssef			
Class UltrasonicSensor	Yousra & Marième			
Class TouchSensor	Anas			
Package 2 : Action		10/28/24	11/3/24	7
Class Déplacement	Anas & Yousra & Marième			
Class Pince	Youssef & Mohammed			
Package 3 : Agent	Groupe	11/4/24	11/10/24	7
Finalisation du plan de developement	Yousra	11/11/24	11/12/24	2
Plan des tests				
Creation des tests unitaires	Groupe	11/13/24	11/14/24	2
Tests d'integration	Groupe	11/14/24	11/16/24	3
Tests de validation	Groupe	11/17/24	11/18/24	2
Code source				
Optimization du code	Groupe	11/18/24	11/19/24	2
Modification du repertoire Github	Groupe	11/20/24	11/20/24	1
Finaliser le Code	Groupe	11/20/24	11/21/24	2
Documentation interne				
Redaction de guide d'utilisation JavaDoc	Groupe	11/21/24	11/21/24	1
Verification et validation de la documentation	Groupe	11/22/24	11/22/24	1
Rapport final				
Synthèse des résultats	Groupe	11/25/24	11/27/24	3
Relecture et correction du rapport	Groupe	11/28/24	11/29/24	2

6. Stratégie d'Optimisation

6.1 Stratégie de Déplacement

Premier palet :
Le robot commence sur la ligne la plus éloignée de l'adversaire, avec un palet positionné devant lui au début. Il commence par récupérer la direction du champ adverse afin de l'utiliser comme référence, puis il identifie le premier palet et se dirige vers celui-ci. Une fois en contact avec le palet, il se sert de son capteur tactile pour valider le contact et ferme ses pinces pour saisir le palet. Par la suite, il modifie sa trajectoire pour se diriger vers le champ adverse. Quand il parvient dans ce secteur, le robot retirera le palet et reculera 10 cm avant de se repositionner adéquatement et de préparer la prochaine action.

Deuxième et troisième palet :
Pour le deuxième palet, le robot avance de 55 cm, effectue une rotation de 90 degrés avant de parcourir la zone pour repérer un palet. Une fois le deuxième palet repéré, le robot avance vers celui-ci et confirme le contact à l'aide de son capteur tactile, puis le conduit dans le champ adverse, où il le dépose. Ensuite, le robot ajuste de nouveau sa position pour chercher le troisième palet. Si le troisième palet est détecté, le robot suit le même processus que pour le deuxième palet.

Autres palets :
Dans le cas où le deuxième ou le troisième palet n'est pas repéré après, ou bien après avoir réussi le dépôt du troisième palet, le robot pivote afin de se positionner dans la direction opposée du camp adverse. Il avance ensuite d'une distance d'environ 120 cm afin d'atteindre une ligne principale qui lui sert d'indicateur. Une fois positionné, il procède à une rotation totale de 360° en utilisant ses capteurs ultrasoniques pour détecter les objets autour de lui.

Lors de cette phase, le robot repère l'objet le plus proche situé à une distance suffisante, supérieure à 34 cm. Il modifie sa direction afin de se rapprocher de l'objet repéré et avance vers celui-ci, tout en vérifiant, durant son déplacement, s'il s'agit d'un palet. Si l'objet est identifié comme un palet, le robot le saisit à l'aide de ses pinces, le transporte jusqu'au champ adverse, le dépose, puis reprend le processus effectué pour les autres palets.

En revanche, si en avançant et en ouvrant les pinces, le capteur tactile n'est pas activé, le robot recule de 40 cm, referme ses pinces et ajuste sa position pour explorer une nouvelle direction. Si l'objet repéré n'est pas un palet ou si aucun objet intéressant n'est trouvé, le robot pivote de 180 degrés, avance de 30 cm, puis enchaîne avec les mêmes étapes réalisées pour autres palets.

Ces étapes de recherche et de détection sont répétées de manière systématique par le robot jusqu'à ce qu'il ait saisi et déposé six palets dans le champ adverse.

6.2 Gestion des Collisions

Le robot utilise un système de détection et d'évitement basé sur un capteur ultrasonique pour surveiller en temps réel l'environnement lors des collisions. Le seuil de détection est fixé à 10 cm, ce qui permet d'identifier les obstacles suffisamment proches pour nécessiter une manœuvre corrective. Une fois qu'un obstacle est repéré, le robot interrompt sa progression afin de mettre en place une procédure d'évitement structurée. Lors de cette procédure, le robot commence par s'éloigner de l'obstacle en pivotant de 90 degrés, puis avance sur une distance de 25 cm, puis effectue une rotation inverse pour contourner l'obstacle. Il modifie ensuite son mouvement en avançant sur une distance équivalente à la mesure du capteur ultrasonique,

augmentée de 10 cm, afin de garantir un dépassement total. Enfin, le robot effectue un mouvement symétrique dans le but de retrouver sa trajectoire initiale et poursuivre sa mission. En combinant ce mécanisme d'évitement avec une surveillance continue des alentours, garantit au robot un déplacement sécurisé dans un environnement dynamique, en s'adaptant efficacement aux imprévus.

7. Tests et Validation

7.1 Tests Matériels

Les tests portes sur les différents capteurs du robot :

- Capteur ultrasons : Nous testons la détection des objets à différentes distances et angles.
- Capteur de couleur : La capacité du robot à suivre une ligne colorée.
- Capteur tactile : Le robot est capable de détecter un contact physique ou une pression lorsqu'il touche un objet.
- Les pinces : Tester la prise et la libération des palets de manière efficace.

7.2 Tests Logiciels

Des vérifications régulières du code pour valider son bon fonctionnement.

- Le test des méthodes :
    - Consiste à s'assurer qu'elles réagissent correctement aux capteurs et renvoient les résultats attendus, tout en garantissant leur bon fonctionnement global dans le système.
- Simulation :
    - Ce test a pour objectif de valider l'intégration cohérente et efficace de tous les composants du programme.

8. Conclusion et Perspectives

Le projet d'intelligence artificielle visant à équiper un robot Mindstorms EV3 pour détecter, manipuler et transporter des palets de manière autonome représente un défi technique et organisationnel majeur. Utilisant Java via l'IDE Eclipse et la librairie LeJOS, nous avons structuré notre approche en plusieurs étapes clés. Les ressources matérielles, telles que le robot EV3 LEGO Mindstorms et les divers capteurs, sont essentielles, mais les limitations des capteurs nécessitent une attention particulière. Les objectifs techniques sont ambitieux : concevoir un robot capable de collecter et déposer un maximum de palets dans un temps imparti, tout en respectant les règles de la compétition. Ce plan de développement offre une vision claire des étapes nécessaires pour atteindre nos objectifs, facilitant la coordination et le suivi du projet.