1. Introduction

Par groupe de 5, nous devons programmer un robot EV3 en langage Java pour que lors de l'épreuve, ce dernier ramasse des palets les uns après les autres et les ramène au fur et à mesure dans notre camp. Cette épreuve se déroule dans un espace restreint, en compétition avec un autre robot placé sur le même plateau de jeu. Ce plateau contient 9 palets, chacun placé à une intersection de deux lignes de couleurs et notre but est de marquer plus de points que notre adversaire en sachant que le calcul des scores s'effectue comme suit :

- 3 points pour chaque palet placé dans la zone d'en-but adverse,
- 5 points de bonus pour l'équipe ayant marqué en premier,
- 2 points pour le robot ayant un palet en sa possession a la fin de la manche.

2. Spécifications fonctionnelles

Cette section doit décrire ce que le système (robot et logiciel) doit faire du point de vue fonctionnel, c'est-à-dire ses principales fonctionnalités et cas d'utilisation. Voici les éléments à inclure :

- **Description des fonctionnalités principales** :

    - Au lancement de l'épreuve, notre robot ramasse le palet situé en face de lui (ouverture des pinces, avance jusqu'à la pression du capteur de toucher à l'avant du robot, puis referme les pinces), et se dirige vers le camp adverse pour le déposer (se décale vers la droite de 45°, avance de 25 cm, tourne vers la gauche de 45° puis avance jusqu'à la ligne blanche du camp adverse).
    - Pour détecter un palet, notre robot recherchera le palet le plus proche de lui en effectuant un tour sur lui-même. Lors de ce tour, notre robot est capable de différencier un mur (données continu dans notre tableau de valeur de distance), un robot adverse (...) et un palet(...) pour conserver uniquement l'emplacement du palet.
    - Ensuite, le robot se déplace en direction du palet observer, l'attrape et le ramène dans le camp adverse (s'oriente grâce à notre boussole en direction du camp adverse puis s'y rend).
    -

    - **Ramassage des palets** : Comment le robot identifiera et ramassera les palets sur le plateau de jeu.
    - **Détection des obstacles** : Capacité du robot à détecter les obstacles (autres robots, murs, etc.).
    - **Navigation** : Comment le robot se déplacera dans le plateau et trouvera les intersections.
    - **Marquage de points** : Le processus pour ramener les palets dans la zone d'en-but et comptabiliser les points.
    - **Réaction aux conditions de jeu** : Réponses du robot face à des événements comme la collision avec le robot adverse.
- **Cas d'utilisation** :
    - **Cas 1 : Début de la manche** : Le robot se déplace vers un palet le plus proche.
    - **Cas 2 : Ramassage d'un palet** : Le robot détecte un palet et le ramasse.
    - **Cas 3 : Conflit avec le robot adverse** : Stratégie du robot pour éviter ou gérer un obstacle (comme l'autre robot).
    - **Cas 4 : Dépôt du palet** : Le robot dépose le palet dans la zone d'en-but.
- **Diagrammes UML (si nécessaire)** :
    - **Diagramme de cas d'utilisation** pour représenter les interactions entre les acteurs (robot, joueur) et le système.
    - **Diagramme de séquence** pour illustrer les étapes d'une tâche précise (comme le ramassage d'un palet).

---

3. Spécifications techniques

Cette partie couvre les aspects techniques et l'architecture du projet. Elle détaille les outils et les technologies utilisés, ainsi que les choix techniques qui influencent le développement.

- **Langages et technologies utilisés** :

    - Pour développer notre projet nous utilisons l'application Eclipse pour programmer en langage Java, l'outil GitHub pour centraliser notre projet et apporter chacune des modifications à notre code et la bibliothèque numérique en ligne de LeJos pour obtenir les différentes informations pratique en lien avec notre robot EV3.

      Notre projet Java s'articule de la manière suivante :

      1 - package Capteur :

      - Classe CapteurDistance :

        // insérer méthodes de la classe CapteurDistance

- Classe CapteurCouleur :

    // insérer méthodes de la classe CapteurCouleur

    - String couleur (int RGB value);

- Classe CapteurContact :

    // insérer méthodes de la classe CapteurContact

    - boolean contact();

- Classe CapteurUltrason :

 // insérer méthodes de la classe CapteurUltrason

2 - package Moteur :

- Classe Pince :

    // insérer méthodes de la classe Pince

    void ouvrir();

    void fermer();

- Classe Roue :

    // insérer méthodes de la classe Roue

    void avancer(int distance);

    void reculer(int distance);

    void tournerDroite(int degre);

    void tournerGauche(int degre);

3 - package IApackage :

// insérer le fonctionnement du robot avec les packages précédent

  - **Java** : Langage de programmation pour contrôler le robot.
  - **EV3 Dev ou LeJOS** : Bibliothèque ou environnement de développement utilisé pour programmer le robot EV3.
  - **Systèmes embarqués** : Technologies spécifiques au matériel du robot (capteurs, moteurs, etc.).
- **Architecture du système** :
  - **Architecture logicielle** : Description de la structure du code (par exemple, séparation en modules pour la navigation, la détection d'obstacles, etc.).
  - **Architecture matérielle** : Schéma des composants matériels du robot (capteurs de distance, pinces pour ramasser les palets, etc.) et leur interaction avec le logiciel.
  - **Diagramme d'architecture** (si nécessaire) : Illustration des différents modules du système et de leur communication.
- **Base de données (si applicable)** :
  - Si ton projet inclut une base de données (pour enregistrer les scores ou les statistiques, par exemple), inclure :
    - **Schéma de la base de données** : Structure des tables, types de données, relations.
    - **Technologie utilisée** : Par exemple, MySQL, SQLite, etc.
- **APIs ou services externes** :
  - **APIs utilisées** (s'il y en a) : Par exemple, si tu utilises des API pour la communication entre plusieurs robots ou pour l'intégration d'un tableau de bord.
  - **Services externes** (s'il y en a) : Par exemple, un service de monitoring ou de contrôle à distance.
-

4. Organisation de l'équipe

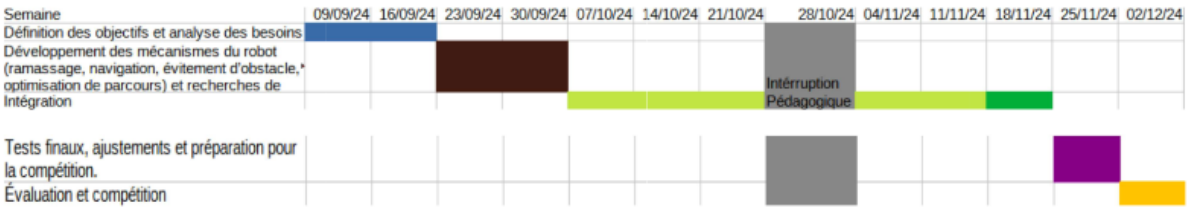
Noms	**Tâches**
Tom	Process
Tao	Cahier des charges
Tao	Plan de développement
Brice	Classe Couleur
Tom	Classe Distance
Adam	Classe Toucher

Brice	Classe Action
	Classe Pince
Brice / Adam / Quentin	Méthode Recherche
Brice / Adam	Classe Principal

- Nous avons mis en place une réunion environ toutes les 3 semaines.

5. Planification

- Échéancier :



- GitHub : https://github.com/AdamInLoveOfBrice/miashsia/

6. Développement et tests

Phases de développement

- Notre code est organisé de la façon suivante : Un package “capteurs” pour gérer les méthodes pour chacun des capteurs, un package “moteurs” pour gérer les actions sur les pinces et sur les roues, et enfin un package principal “iapackage”.

Stratégie de tests

- Tests unitaires sur chaque module (capteur de distance, couleur, pince, etc.).
- Tests d'intégration pour valider la coordination des différents modules.
- Tests en conditions réelles (plateau de jeu) pour ajuster les comportements.

Gestion des bugs et des retours utilisateurs

- Mise en place d’un journal des anomalies pour documenter les bugs.
- Validation des corrections via des tests réitérés.
- Feedback de l’équipe sur l’expérience utilisateur pour affiner les fonctionnalités.
-

7. Livrables et documentation

Liste des livrables attendus

- Cahier des charges
- Plan de développement
- Plan de tests
- Code source complet et structuré.
- Rapport final

Manuel d'utilisateur

- Explication simple des fonctionnalités principales.
- Guide étape par étape pour configurer et utiliser le robot.

Documentation technique pour les développeurs

- Schémas UML des modules.
- Description de l’architecture logicielle et des algorithmes utilisés.

---

8. Suivi et maintenance

Suivi des performances

- Analyse périodique des performances du robot en compétition.
- Mesure de la précision des capteurs et de l’efficacité des déplacements.

Plan de maintenance et d’évolution

- Révision régulière du code pour l’optimiser.
- Ajout de nouvelles fonctionnalités pour s’adapter à des règles ou défis différents.

Gestion des versions et mises à jour

- Utilisation d’un système de contrôle de version (Git).
- Documentation des modifications pour chaque version (changelog).

---

9. Conclusion

Récapitulatif des objectifs atteints

- Développement d’un robot capable de ramasser et de ramener des palets en respectant les spécifications du projet.
- Validation des fonctionnalités principales grâce à des tests en conditions réelles.

Pistes d'amélioration ou d'extension future

- Optimisation des algorithmes de déplacement pour une vitesse accrue.
- Amélioration de la reconnaissance d’objets pour différencier plus précisément les palets des autres éléments du plateau.
- Possibilité d’intégrer des stratégies plus avancées en compétition.
-