Projet IA & Robotique

Plan de développement

Tuteur:

DAMIEN Pellier

Étudiants:

AURAY Cédric

CALVIGNAC Sébastien

SIMON Dorian

KAMISSOKO Djoko

Année scolaire : 2020-2021

Table des matières

1. DESCRIPTION FONCTIONNELLE

a. Classe Agent_Final

i. tempsTotalSec

La méthode tempsTotalSec() est une méthode qui permet d'avoir une horloge interne permettant de mesurer le temps écoulé depuis le début de l'épreuve. On prendra le temps courant du système (en millisecondes) qu'on soustrait à la valeur du temps de départ (topDepart). Il faudra qu'on fasse attention aux unités obtenues et demandées. Pour éviter les erreurs, on définit un format qu'on utilisera tout le long.

ii. getBut

La méthode getBut() est une méthode qui retourne la valeur de l'attribut but.

iii. deplacementAleatoire

La méthode deplacementAleatoire() est une méthode qui permet de se déplacer en fonction du temps. Elle permet de se déplacer durant un temps aléatoire entre 0 et 10 secondes. Ensuite on effectue une rotation aléatoire en 0 et 360°. Durant ce déplacement en ligne droite, le robot interroge son sonar en continu pour savoir s'il détecte un objet à moins de 20 centimètres. Si oui, on effectue un demi-tour avant de quitter la méthode. Si le capteur de pression est pressé lors de ce déplacement sans intention de prise de palet, on serre les pinces et l'on met à jour l'attribut prisepalet.

iv. chercherPalet

La méthode chercherPalet() est une méthode qui fait effectuer au robot des rotations successives jusqu'à ce qu'on détecte un objet dans une zone suffisamment proche. On définit la zone de recherche entre [0.4 ; 0.8]. De plus, si le nombre d'itérations dépasse 36 on quitte la méthode (36 itérations représentant un tour complet). Retourne true si un objet est détecté, mise à jour de l'attribut paletDetecte.

v. prendrePalet

La méthode prendrePalet() est une méthode qui permet de nous saisir d'un palet détecté. Le robot se déplacera en ligne droite tant que le capteur de butée n'est pas activé. Lorsque celui-ci est activé les pinces se referment. Le code retourne true si le palet est effectivement saisi. La méthode retournera false si le robot détecte une ligne blanche ou une distance inférieure à 20 centimètres.

vi. marquerPalet

La méthode marquerPalet() est une méthode qui permet de déposer le palet saisi derrière la ligne d'en but adverse. Cette méthode inclut le recours à l'historique des rotations effectuées pour déterminer la bonne orientation pour se déplacer vers l'en-but cible, ainsi qu'une vérification permettant de nous assurer que nous transportons bien le palet derrière la ligne adverse. La vérification se basera sur la dernière ligne de couleur verte ou bleue perçue. Si la dernière couleur perçue est la couleur qui se trouve devant notre ligne blanche on fait demi - tour. Pour s'assurer que notre robot ne reste pas inactif on lui donnera un temps maximal d'exécution (60 secondes). On s'assure que le palet est bien marqué en renvoyant un booléen.

vii. marquerPalet2

La méthode marquetPalet2() est une méthode permettant de marquer lorsque marquerPalet est en échec. Le fonctionnement est identique à cela près que l'on ne considère plus l'historique des rotations mais que nous évoluons par des appels successifs à chercherBut() jusqu'à discerner un ligne blanche. Si celle-ci

passe le système de vérification sur l'historique de la dernière ligne traversée, l'on marque et l'on renvoie true.
Si l'on arrive pas à cet objectif prévu dans un temps de 60 secondes, arrêt de la méthode et renvoie false.

viii. chercherBut

La méthode chercherBut() est une méthode au fonctionnement similaire à deplacementAleatoire() à ceci près que l'on recherche une ligne blanche. Si celle-ci est perçue et que la vérification conditionnelle d'après la dernière ligne de couleur traversée est validée, l'on a trouvé la zone d'en-but et l'on renvoie true, sinon l'on effectue un demi-tour et l'on poursuit l'execution. Si l'en-but n'est pas trouvé dans un laps de temps de 60sec, l'on arrête l'exécution et l'on renvoie false.

ix. Sequence1

La méthode sequence1() est la première séquence lancée par le robot pour récupérer le premier palet. Il s'agit d'avancer jusqu'à entrer en contact avec un palet puis d'aller dans la zone d'en-but tout en évitant de faire contact avec les autres palets de la même ligne. Cette séquence fait appel à méthode l'homologation.

x. sequence2

La méthode sequence2() fait appel à deplacementAleatoire() et rechercherPalet() pour essayer de trouver des palets sur la table. Les déplacements étant aléatoires, on s'attend à ce que le robot puisse faire des recherches de palets sur une grande partie de la table (et non pas la même zone). Ces déplacements doivent être capables de prendre en compte son environnement, notamment pour éviter les murs.

xi. sequence3

La méthode sequence3() est appelée lorsque le robot a perçu un palet. Cette méthode permet d'aligner la trajectoire du robot avec le palet. Il doit être capable de gérer le cas où il y a échec de prise du palet.

xii. Sequence4

La méthode sequence4() permet de retourner le palet saisi à la zone d'en-but. Elle repose sur le fait que l'historique des rotations a été stocké en attribut dans pilote. Cette méthode doit être capable de gérer le cas où il y a échec du retour du palet, notamment lorsque l'historique des angles de rotation contient un cumul d'erreurs trop importantes.

xiii. Homologation

La méthode Homologation() est une méthode qui avance le robot jusqu'au premier palet. Elle avance jusqu'à récupérer un palet, le saisit et effectue une légère rotation, puis avance jusqu'à l'en-but adverse et dépose le palet. S'il échoue, il abandonne la méthode après une rotation de 180°.

xiv. updateEtat

La méthode updateEtat() est une méthode qui met à jour l'état du actuel du robot d'après les attributs d'instance, l'état courant est référencé comme une chaîne de caractère stockée à l'indice 0 de l'attribut etat qui est un tableau de String.

xv. afficheEtat

La méthode afficheEtat() permet d'afficher sur la brique du robot une chaine de caractère représentant l'état actuel actuel dans lequel le robot se trouve.

xvi. retourneSequence

La méthode retourneSequence() est une méthode permettant de retourner la séquence appropriée en fonction de l'état courant de l'agent.

xvii. Calibrer

Cette méthode permet de restaurer à 0 l'attribut rotation de la classe Pilote. Pour ce faire, on va mettre le robot perpendiculaire au mur. On saura que nous sommes perpendiculaires au mur quand la distance mesurée face au mur sera la plus faible. On cherchera ce point en effectuant des rotations de 5° successives. Après chaque rotation, on mesure la distance et on la compare à la précédente.

b. Classe Capteurs

i. Constructeur Capteur

Le constrcteur de la classe Capteurs permet de créer une et unique instance de Couleur, Touche, Ultrason et d'initilialiser les attributs touche, distance, et couleur en récupérant ces valeurs sur les attributs de ces instances précédemment crées.

ii. actualise

La méthode actualise() permet de lancer actualise() sur les 3 instances que sont Couleur, Touche et Ultrason dans le but de mettre à jour leurs attributs, puis de récupérer ces valeurs pour les assigner aux attributs de l'instance courante de Capteurs.
L'instance de Capteurs qui sera créée et aura alors pour rôle de centraliser les données en provenance des différents capteurs du robot.

c. Classe Couleur

i. Constructeur Couleur

Le constructeur permet d'initialiser un port matériel sur la brique et de l'associer au capteur couleur ; mais également de réaliser un calibrage (nécéssité de porter manuellement le capteur sur les objets physiques de couleur donnée pour enregistrer les données archétypes dans les tableau correspondants - 3 éléments par tableau correspondant aux valeurs RGB) ;une instance de SampleProvider se charge de récupérer les données perçues par le capteur, qui sont ensuite copiées dans les objets dédiés.

ii. actualise

La méthode actualise() permet quant à elle de récupérer un échantillon, pour ensuite le comparer aux valeurs archétypes (tableaux en attributs) et modifier l'attribut String couleur en conséquence pour permettre de récupérer cette valeur à l'extérieur de l'instance courante. Pour ceci l'échantillon est comparé successivement aux différents archétypes de couleur successivement pour rechercher celui qui correspondra à la plus petit erreur possible.

iii. scalaire

La méthode scalaire qui prend en paramètre le tableau contenu l'échantillon perçue et le tableau d'un archétype de couleur (bleu, rouge, vert etc.), permet de calculer la différence entre les deux et la retourner sous le format d'un double.

d. Classe Ultrason

i. Actualise

La méthode actualise() permet de mettre à jour cette valeur d'attribut en effectuant un nouvel échantillon de données.

e. Classe Touche

i. Constructeur Touche

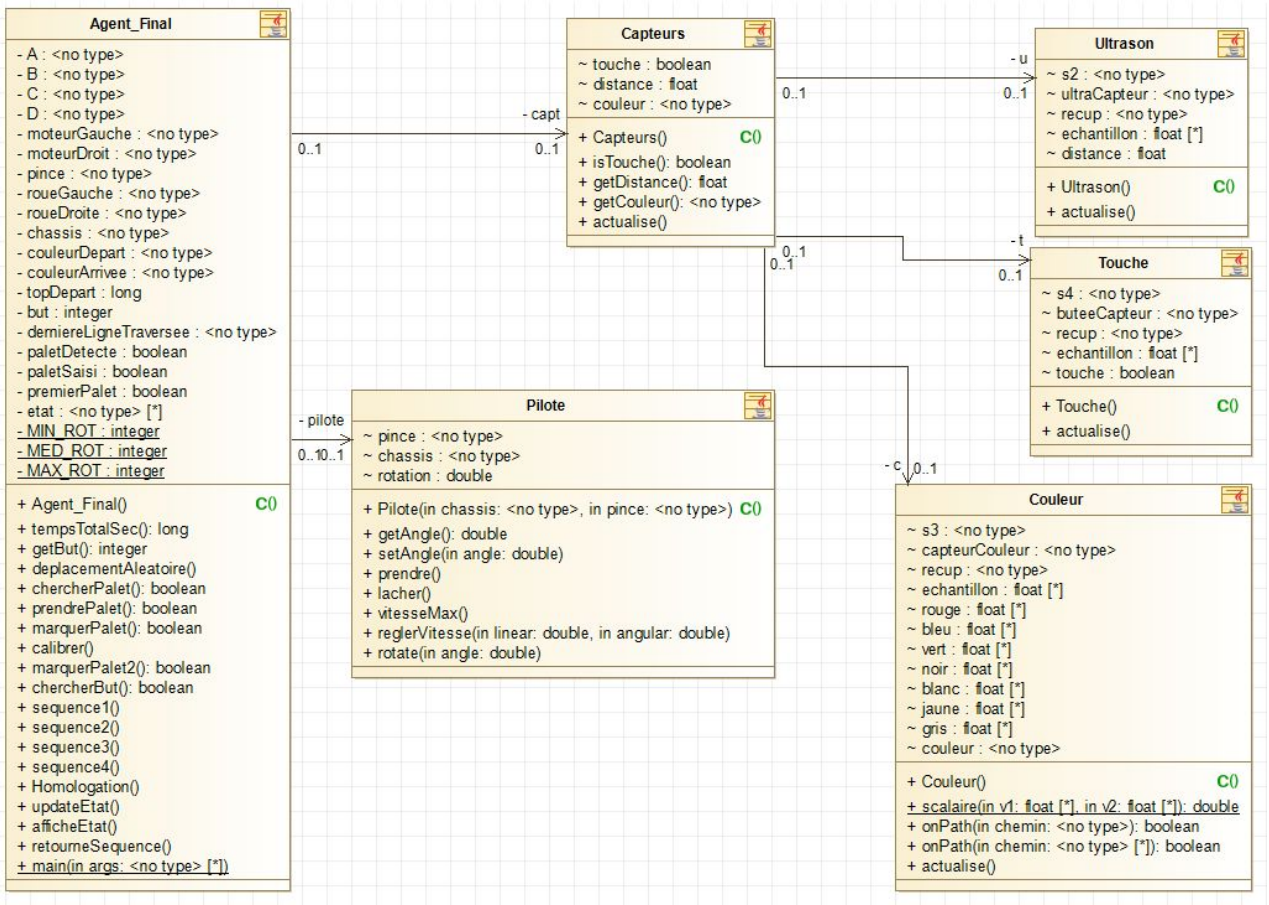
Le constructeur permet d'initialiser un port matériel sur la brique et de l'associer au capteur de butée. Celui-ci stocke la valeur correspondant à une pression perçue ou non. L'échantillon est effectué par une instance de SampleProvider et stocké dans un tableau de taille égale au tableau d'échantillon (1 élément). Puis cette valeur est convertie en boolean (attribut touche ; true si 1, false si O) via un appel à actualise().

ii. Actualise

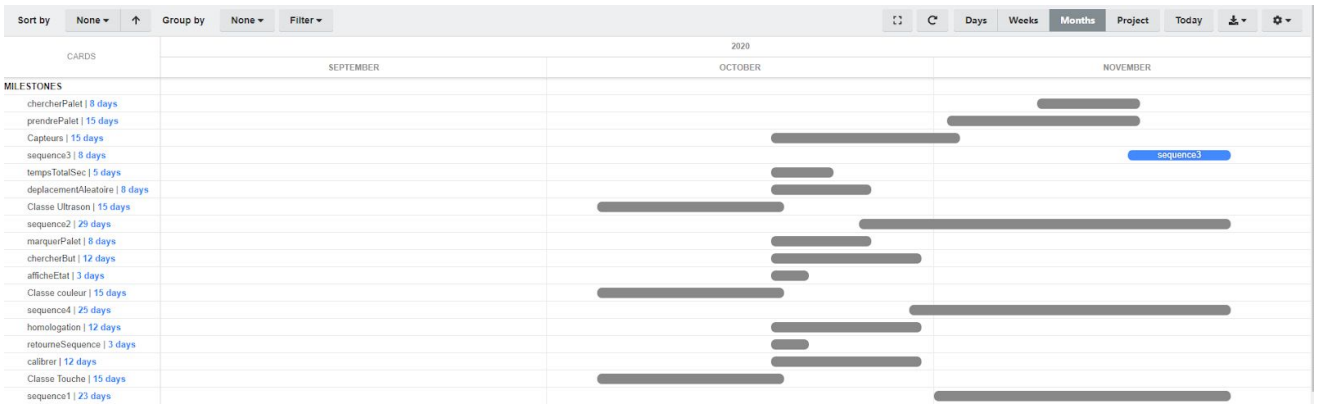
La méthode actualise() permet de mettre à jour cette valeur d'attribut en effectuant un nouvel échantillon de données.

3. DIAGRAMMES

a. UML



b. Diagramme de Gantt



c. Répartition des tâches