Rapport de la séance 14

28 février 2022
PORCEL Koralie
Robotique

1) Code capteurs de distance

Lors de cette séance, j'ai fait le code pour la détection d'obstacle sur Arduino.

J'ai rajouté trois fonctions dans la classe Moving qui permet à l'araignée de se tourner à gauche puis à droite puis revenir au milieu.

J'ai créé une nouvelle classe Capteurs. Dans cette classe, il y a :

- *capteurIR()* : qui renvoie la distance de détection d'un obstacle à l'aide du capteur à infrarouge LIDAR TF-Luna
- *capteurUltrason()* : qui renvoie la distance de détection d'un obstacle à l'aide du capteur à ultrasons HC-SR04
- *mouvementVerrifObstacle()* : permet à l'araignée de se tourner à 40 ° sur elle-même à gauche puis à droite et appelle à chaque fois la fonction *distanceObstacle(indice),* l'indice est 0 la 1er fois qu'il tourne, puis 1 et enfin 2. A la fin de la fonction, appelle *actionSiObstacle()*
- *distanceObstacle(int indice)* : appelle les fonctions *capteurIR()* et *capteurUltrason()* et vérifie s'il détecte des obstacles à moins de 50 cm (valeur à changer). Si c'est le cas, mets true dans un tableau à l'indice indice, sinon mets false
- *actionSiObstacle()* : analyse le tableau et réagie différemment :
  Obstacle à gauche : tourne de 45° à droite
  Obstacle à droite : tourne de 45° à gauche
  Obstacle à gauche et milieu : tourne de 90° à droite
  Obstacle à droite et milieu : tourne de 90° à gauche
  Obstacle au milieu : tourne de 90° à droite
  Obstacle à gauche et droite : tourne de 135° à droite
  Obstacle à gauche droite et milieu : tourne de 135° à droite
  Aucun obstacle : ne fais rien

La fonction *mouvementVerrifObstacle()* sera appelée toutes les 8 secondes en utilisant millis pour vérifier les obstacles. Toutes les secondes, vérifiera s'il n'y a pas d'obstacle quand l'araignée se déplace.

Lors des testes que j'ai fait, j'ai remarqué que j'ai du baissé l'angle auquel l'araignée tourne sur elle-même de 45° à 40° car sinon les capteurs captaient parfois les pattes de l'araignée. Je vais continuer à réfléchir à quelle distance est optimal pour se tourner

lorsque l'araignée détecte un obstacle. Et peut-être rajouter d'autres cas s'il détecte un obstacle trop proche, recule avant de tourner.

2) Code déplacement araignée

J'ai amélioré le code dans la classe Moving pour déplacer l'araignée. Les fonctions *avancer()* et *reculer()*.

Au lieu de créer deux fonctions pour séparer le mouvement en deux du déplacement, je remplace cela par une seule fonction.

Dans cette fonction, j'ai deux listes : listeAddition1 et listeAddition2 qui va incrémenter pour chaque déplacement la listePosition à l'aide de plusieurs cas. Après chaque incrémentation, l'araignée va aller à la position dite dans la listePosition :

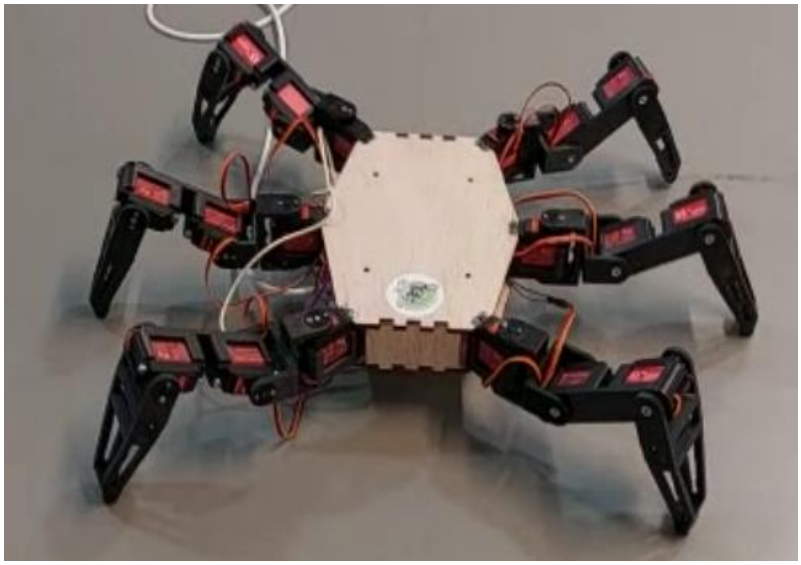
- Si les pattes 1,3 et 5 touchent le sol (ces trois pattes marchent de pair) et les pattes 2, 3 et 5 sont en l'air alors : incrémente la listePosition avec la listeAddition1 car le mouvement est modifié de +listeAddition1 entre chaque mouvement
- Si les pattes 1,3 et 5 sont en l'air et les pattes 2, 3 et 5 touchent le sol alors : incrémente la listePosition avec la listeAddition2

Pour savoir si la patte est en train de se lever du sol ou bien de se rapprocher, j'utilise plusieurs cas :

- Si la position de la patte verticale 1,3 et 5 sont au max, ou la position des pattes 2,4 et 6 sont au max alors la patte va ensuite devoir redescendre.
  Pour montrer cette modification, je change une variable teta et beta qui sont dans les listes listeAddition1 et listeAddition2 (ici teta et beta sont positifs)
- Si toutes les pattes sont au sol alors la prochaine va devoir monter (ici je change teta et beta en leurs valeurs négatives).

Avec le même principe, j'ai fait la fonction reculer.

Faire cela permet de pouvoir faire s'arrêter l'araignée avec moins de latence (ne doit pas attendre que la fonction complète du déplacement soit terminée pour pouvoir l'arrêter), avoir un code plus propre. Si elle doit s'arrêter pour tourner après la détection d'un obstacle, elle se remet dans la position normale. Pareil si elle doit commencer à se déplacer :



1) position dite normale

Je vais ensuite améliorer le code de la rotation d'une manière similaire.