Compte-rendu du TME 8 :

- Grosse prise de retard car on est parti sur la balise sans avoir clôturé ce qu'il faut pour prendre le robot en main, le contrat n'est pas rempli, le proxy doit être notre priorité absolue

- Il faut écrire le temps prévu et le temps mis pour chaque tâche dans le Trello maintenant

- On a pas fait de séparation entre module et sous-modules : on a tout dans module, dur de s'y retrouver

- Le main de projet.py fait 30 lignes : c'est trop, dans le main il faut juste créer une IA, le robot, l'arène. Les imports de module etc. doivent être dans un autre fichier, pas de threadrun ni de stratégies
→ on peut faire un core dans module qui sert de sous-main où se passent les initialisations

- L'affichage 2D prend un robot en paramètre, à la fin ce sera un proxy à la place

- La classe balise et les fonctions detect et capture ne devraient pas être dans controleur.py

- Le code de strategieSuivreBalise devra être réécrit car la balise appartient au robot mais surtout on a pas fait le proxy : certaines choses de la stratégie devront appartenir au proxy

- Les controleurs font appel à getX&Ystep(dt) du robot alors que robot.x et robot.y n'existent pas dans la réalité

- On a toujours pas vu les difficultés qui se cachent derrière le proxy → difficultés en vue : est-ce au controleur ou au proxy de mettre à jour ? Où reporter le cumul de la distance parcourue ?

- Problèmes au niveau du dt : passer le dt à partir de la stratégie Avance n'est pas une bonne idée, ça ne devrait pas être une constante, c'est à la méthode de savoir combien de temps s'est écoulé depuis son dernier appel

- Les IA doivent être le plus légères possibles, les calculs de distance etc. ne doivent pas se faire dans l'IA mais dans le proxy, elles n'ont le droit d'appeler que les fonctions du proxy

- Le robot réel n'a que 4 méthodes : setMotorDps (fixer la vitesse), getPositionMotor(savoir de combien il a tourné), getDistance(savoir la distance au mur) et offsetMotorEncoder (remettre à zéro sa distance relative)

- Notre simulateur a beaucoup plus de méthodes que le robot, il faut que le simulateur se comporte comme le robot → Il ne faut garder dans le simulateur que les méthodes qui existent dans le robot, et mettre le reste dans le proxy, autrement dit toutes toutes les fonctions de la partie simulation que l'on aura pas quand on aura accès au robot réel vont dans le proxy

- On doit donner des fonctionnalités supplémentaires au robot réel grâce au proxy, comme calculer la distance parcourue et lui donner des coordonnées

- Mettre des pass dans le corps des fonctions de robot_reel.py pour pouvoir l'exécuter (ne renvoie rien mais permet de détecter des erreurs de compilation etc.)