Structure of the mecatronics programm

**Main structure :**

1. Init (only once):
   1. initialize all the value to begin the program (can, spi, struct, …)
   2. puts all the motors ctrl to 0, 1
   3. init Map and Path
2. Infinite loop (keeps being called until stops):
   1. Update info (pos ctrl)
      1. Position
      2. Motor speed
      3. Beam detection
   2. Compute cmd (cfr ia) – High level controller
      1. Switch: state? (action described below) -> update wheel speed wanted
   3. Action: (for now just one function)
      1. Run\_speed\_controller : cmd sent to the motors
   4. Other fctns: (for now just one function)
      1. Write data to files

**State actions:**

1. Calibration:
   1. … to be done
   2. Detect wall -> Initpos (function that initialize on of the coordinate)
   3. … to be done
   4. Change state to (4 or 2 or …)
2. Action (manche à air, phare, …)
   1. …
3. Navigation
   1. Calculate distance toward objective (stored in path)
   2. If dist == 0 => change obj (path[i+1])
   3. Else middle\_level\_ctrl : update wanted speed to achieve dist and orientation
4. Test (free state to be able to make test)
   1. …
5. Stop
   1. Wanted speed to 0
   2. Change state (could be interresting to create a fct that check if the state has to be changed)

Rmq : on pourrait programmer pour que à a fin de chaque state (genre calib finie ou action finie ou …) il doive retourner en state 4 avant de rechanger de state (à voir …)

Rmq : il y aura peut-être un state « avoidance » pour lors ce qu’on rencontre un autre robot (à voir comment on programme ça)

**Path planning**

Voir code et faire des commentaires

Rmq : si nos trajectoires sont calculées de manière rectiligne (pour le path planning et pour la navigation) il faut faire tourner robot (vers objectif) -> avancer tout droit -> tourner robot (pour régler son orientation) probablement moyen de penser pour pouvoir avoir des trajectoires plus intelligentes

Rmq : réfléchir pour la gestion d’une action qui ne se fait pas sur un point fixe (faire tomber les manches à air)

Rmq : le path planning ne se fait pour l’instant que de nœuds à nœuds, peut être besoin d’implémenter une consigne si hors des nœuds

Rmq : le path planning et le middle level controller ont été implémentés de manière que le robot fasse des arcs de cercle avec un angle constant de max pi/8 donc dans le choix des nœuds il faut vérifier si tous les nœuds sont accessible en utilisant cette courbe + tous les nœuds doivent être accessible d’une seule traite depuis la base (pour facilité de l’implémentation de pathplanning)

**Robot avoidance**

**…**

**En plus**

* Avoidance du Robot pourrait se faire à n’importe quel moment (via un interupt lancé par le De10 dans la main)
* Dans la fonction position de updateinfo un calcul de l’erreur (proba cfr cours de robotique) peut-être mis à jour et une recalibration (le robot passe en state « calibration ») peut être faite dans les coins si « l’erreur probable est trop grande »
* Un recalcul du path planning peut être refait quand on veut (à priori c’est un calcul lourd donc à éviter mais peut-être nécessaire après une phase d’avoidance de robot adverse)