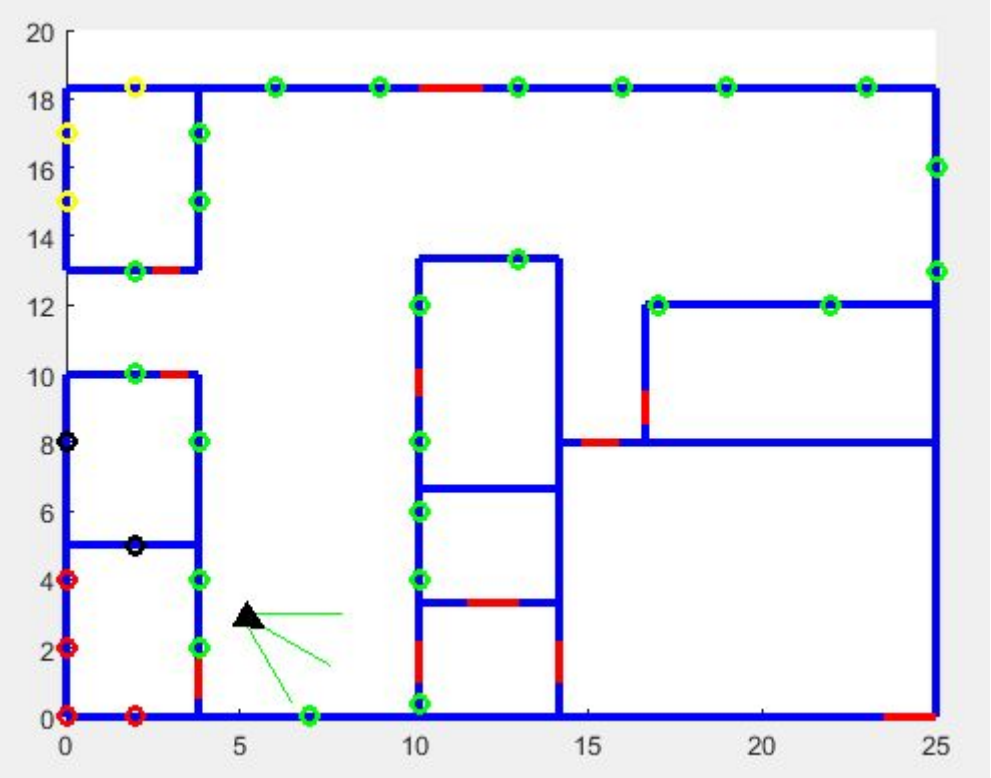
Documentation Cartographie

I. Cartographie



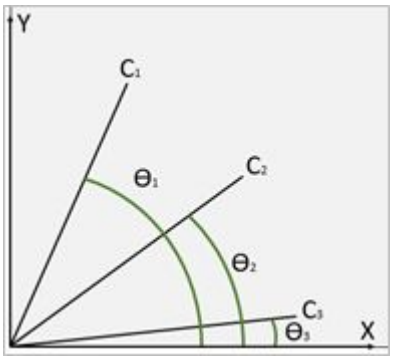
Dans cette partie, nous avons pour mission la représentation du bâtiment 14 sous matlab. Dans un premier temps, nous avons récupéré sur un plan, l'ensemble des mesures importantes. Nous avons récupéré la longueur et la largeur du bâtiment. Ensuite, nous avons récupéré les points importants du bâtiment, correspondant aux coins de chaque salle approximativement.
Nous avons alors une représentation du bâtiment dans son ensemble, avec les traits bleus représentant les murs et les traits rouges les portes. A ce stade, on peut affirmer que les mesures réalisées sur le plan ont une erreur au millimètre, donc avec l'échelle du plan, on peut affirmer que l'erreur de position de l'ordre de la dizaine de centimètre. Nous avons ensuite placé des tags, qui sont séparés en plusieurs catégories : en vert les tags présents dans l'espace opérationnel, en jaune les tags dans le bureau de Dubreuil, en rouge les tags dans le bureau de François et enfin en noir ceux présents dnas la salle de réunion.
Dans la partie suivante, celle traitant du simulateur, nous expliquerons comment le robot se déplace dans l'espace opérationnel, pour atteindre un tag désiré.

II. Simulateur

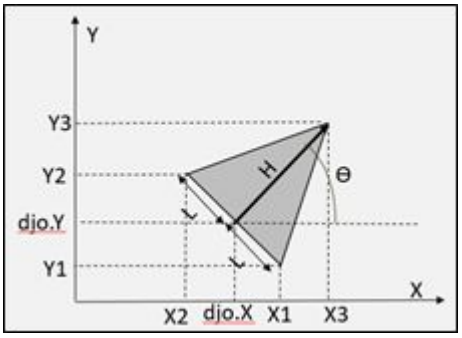
ROB.INIT :

Dans notre programme, le robot est défini par une structure appelée « djo ». Elle contient toutes les dimensions, valeurs et coordonnées le caractérisant. On y retrouve toutes les valeurs du cahier des charges. On initialise les variables correspondant aux vitesses angulaires des roues « djo.wd » et « djo.wg ». Pour finir, on initialise la variable «

djo.dist_fin_max » à 0.05. Elle correspond à la distance maximale autorisée entre le robot et l'arrivée pour lui permettre de s'arrêter si l'angle d'arrivée voulu est également respecté.



Sur ce schéma, chaque trait correspond au laser d'un capteur. Le capteur C2 est confondu avec le vecteur directeur du robot. Par conséquent, son angle dans le repère de la figure vaut « djo.theta ». De plus, les deux autres lasers sont séparés d'un angle π/6 avec le laser du milieu C2. Par conséquent, l'angle de C1 vaut « djo.theta »+π/6 et celui de C3 vaut « djo.theta »-π/6. Chaque distance des lasers sont initialisées à « djo.porte », soit 3. De plus, nous avons créé une matrice « couleur » dans la structure qui permet tout simplement d'avoir la possibilité de changer la couleur de ces lasers si nécessaire. Pour finir, on affiche les capteurs grâce à la fonction « line », que l'on implémente également dans une variable pour pouvoir modifier les coordonnées si nécessaires.



Pour pouvoir afficher le dessin du robot, on utilise une fonction nommée « patch » prenant en argument la matrice « vertices_djo » contenant les coordonnées des sommets de la figure. Elle prend aussi la matrice « face_djo » dont la taille correspond au nombre de sommets et qui contient tout simplement les numéros attribués à chaque sommet. Ainsi, les premières coordonnées de la matrice « vertices_djo » correspond au sommet dont le numéro est donné par le premier élément de « face_djo ». En utilisant le schéma « figure 2 », on obtient facilement les coordonnées des trois points.

TAG :
Cette fonction permet d'afficher tous les tags et renvoie une matrice 32x2 contenant les coordonnées des tags. Cela permet de remplacer la base de données initialement prévue.

SIMULATEUR :
Dans cette fonction, nous réalisons le déplacement du robot. Premièrement, on demande à l'utilisateur de rentrer le tag detecté puis le robot part dans sa direction. Après la distance minimale entre le robot et le point d'arrivée fixé, le robot s'arrête. La fonction set permet de déplacer les dessins dont les pointeurs sont mis en paramètre, avec les nouvelles coordonnées. Celles-ci sont calculées par la fonction ODE45 qui permet de résoudre l'équation différentiel qui se trouve dans la fonction MODELE. La variable de commande est calculée par la fonction CONTROLE.

CONTROLE :
Dans cette fonction, nous réduisons la vitesse des roues lorsque le robot se rapproche du point d'arrivée. Nous avons également ajouté les modifications de vitesse en fonction de la détection des capteurs. Il faut cependant ajouter dans le code une fonction permettant de calculer la distance de détection.

CE QU’IL FAUT AJOUTER:

- Un code permettant de calculer la distance de détection des capteurs
- Modifier la fonction CARTOGRAPHIE pour renvoyer les coordonnées des murs
- Ajouter une fonction permettant de choisir un point d’arrivée proche du tag choisi (ou de plusieurs tags)
- Lier le code de la STM et le code MATLAB pour avoir un déplacement en directe en fonction des tags détectés