AZZOPARDI

Thomas

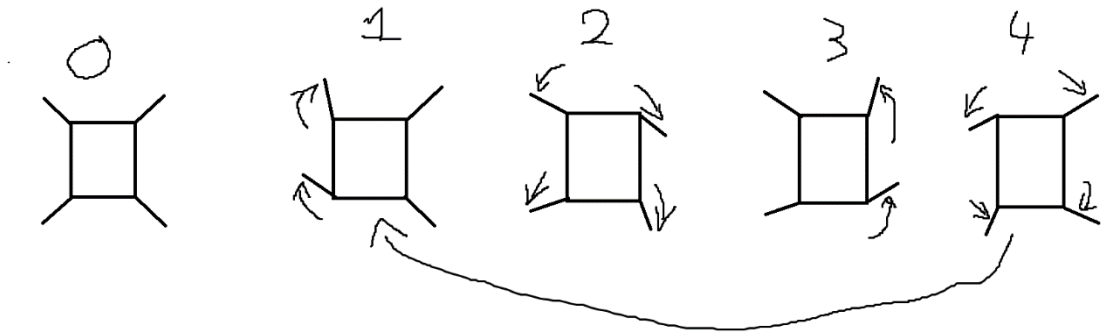
Peip 2 Rapport de séance n6

Après avoir assemblé complètement, j'ai pu enfin faire des tests concrets sur notre Quadrupède. Cette fois la Fonction MoveF (qui permet de faire avancer notre quadrupède) à été réalisé en pensant au 4 pates :

Pour commencer, Il a fallu revoir la fonction « déplacement » qui permettait de faire une rotation « stable » d'un servo-moteur (en rajoutant un délai). Maintenant celle-ci prend un paramètre en plus : « avancement » qui est soit vraie soit faut. Si ce paramètre est vrai, on Rajoute l'angle au servo-moteur, sinon on le retire. Voici la fonction :

```
void Deplacement(Servo moteur,int time,int Ini,int angle, bool avancement){
  if (avancement==true){
    for (int i = Ini; i<Ini+angle;i++){
    moteur.write(i);
    delay(time);
    }
  }
  else {
    for (int i = Ini; i>Ini-angle;i--){
    moteur.write(i);
    delay(time);
    }
  }
}
```

Pour poursuivre, il nous fallait un algorithme pour savoir Comment notre quadrupède se déplacera-t-il ? Pour ceci nous nous sommes inspiré de la nature en elle-même pour en déduire qu'elle se déplacera selon l'algorithme suivant (réduisant nos contraite d'équilibre car les pates ne partent pas dans tous les sens) : après l'initialisation faites, on répète l'algorithme de l'étape 1 à 4.



L'étape 1 se fait à partir de la fonction « placement » qui place une pate à l'angle souhaité.

```
void placement(Servo moteur1, Servo moteur2,int angle){
  Deplacement(moteur1,5,Angle_Ini,angle,true);
  Deplacement(moteur2,5,Angle_Ini,angle,true);
  Deplacement(moteur1,5,Angle_Ini+angle,angle,false);
}
```

Celle-ci utilise la fonction « Deplacement » vu précédemment.

Pour faire avancé notre Quadrupède (étape 2 et 4) il faut « poussé » sur les pates. Voila d'où nous ai venu de faire la fonction « pousse » qui sera la partie d'avancement concrète de notre Quadrupède.

```
void pousse(Servo moteur1,Servo moteur2,Servo moteur3,Servo moteur4,int angle_ini,int angle){
  Deplacement(moteur1,5,Angle_Ini,angle,false);
  Deplacement(moteur2,5,Angle_Ini,angle,false);
  Deplacement(moteur3,5,Angle_Ini,angle,false);
  Deplacement(moteur4,5,Angle_Ini,angle,false);
}
```

Enfin, après avoir tous les outils, nous avons créé la fonction « MoveF » qui permet d'avancer tout droit. Celle-ci reprends toutes les fonctions précédentes pour faire fonctionner notre quadrupède.

Qui reproduit le schéma de notre algorithme.

```
//on suppose que on commence à 90 degrès soit en étoile
void moveF(Servo AvDH, Servo AvDB ,Servo AvGH, Servo AvGB,Servo ArDH,Servo ArDB,Servo ArGH,Servo ArGB){          // fonction pour aller vers l'avant
  //on commence par le double placement à droite
  placement(AvDB,AvDH,30);
  placement(ArDB,ArDH,30); //théoriquement on est  à 130°
  //on pousse pour avancer :
  pousse(AvDH,AvGH,ArGH,ArDH,Angle_Ini+30,60); //on pousse jusqu'à -40 de l'initial
  //on place à gauche
  placement(AvGB,AvGH,60);
  placement(ArGB,ArGH,60); // on était à 50 -> 120
  //on pousse
  pousse(AvDH,AvGH,ArGH,ArDH,Angle_Ini+30,60);

}
```

CONTRAITE :

Pas tout les servos-moteurs vont dans le même sens : ceux à droite vont dans le sens opposé de ceux à gauche. Il a fallu donc réfléchir en fonction de cela.

L'angle initial reste le milieu de tout les servo-moteur afin d'être plus efficace lors de ces mouvements (90 degrés).

Pour terminer, nous nous sommes rendu compte que les pates n'accrochaient pas sur notre table ou autres surfaces. Nous avons donc pensé qu'il fallait les modifié et rajouter une matières qui accroche plus au sol.

J’ai donc modifié les pates pour y faire une sphère pour certainement y rajouter / coller une matière qui adhère plus efficacement au sol.