

Bat Monkey

Rapport de la septième séance(21 février) :

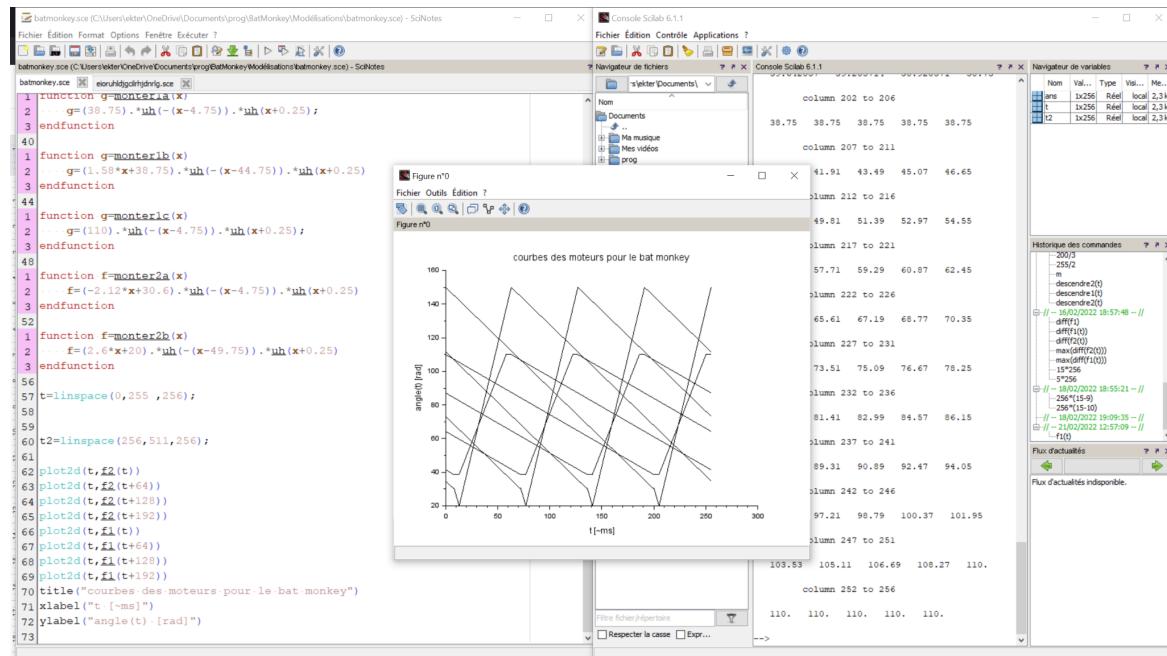
Pendant les vacances, j'ai créé l'application de commande du Bat Monkey sur MIT App Inventor. Je n'ai pas pu faire les tests tout de suite par manque de fils, mais j'ai écrit le programme pour l'analyse des informations envoyées par le BatMonkey dans une trame contenant les informations comme les angles des moteurs ou les codes d'erreur séparées par des virgules.

Voici un exemple de trame : [37,74,111,25,19,13,1,2,255,5,0]

Les 8 premiers nombres sont les inclinaisons des moteurs, viennent ensuite l'étape, la vitesse et le code d'erreur.

J'ai aussi fait des fonctions pour renforcer la sécurité des moteurs, et vérifier à tout moment si l'angle qu'on leur demande est trop éloigné ou s'ils sont débranchés par exemple.

L'objectif ultime serait un robot qui reste accroché à l'arbre même si le courant est coupé, mais c'est impossible avec les servomoteurs, alors nous nous contenterons d'un programme qui peut recommencer à tout moment sans que les moteurs en soient impactés et qu'ils ne reprennent pas tout de 0. Ce programme n'est pas encore fait, mais il est désormais à portée de code grâce aux nombreuses fonctions écrites pour la position des bras, modélisées sur Scilab.



```
batmonkey.sce (C:\Users\ektor\OneDrive\Documents\prog\BatMonkey\Modifications\barmonkey.sce) - Scilab Notes
Fichier Edition Format Options Fenêtre Exécuter ?
batmonkey.sce [en cours] g-monster1(x)
1 function g=monster1(x)
2   g=(38.75).*uh(-(x-4.75)).*uh(x+0.25);
3 endfunction
40
1 function g=monster1b(x)
2   g=(1.58*x+38.75).*uh(-(x-44.75)).*uh(x+0.25);
3 endfunction
44
1 function g=monster1c(x)
2   g=(110).*uh(-(x-4.75)).*uh(x+0.25);
3 endfunction
48
1 function f=monster2a(x)
2   f=(-2.12*x+30.6).*uh(-(x-4.75)).*uh(x+0.25);
3 endfunction
52
1 function f=monster2b(x)
2   f=(2.6*x+20).*uh(-(x-49.75)).*uh(x+0.25);
3 endfunction
56
57 t=linspace(0,255,256);
58
59
60 t2=linspace(256,511,256);
61
62 plot2d(t,f1(t))
63 plot2d(t,f2(t+4))
64 plot2d(t,f3(t+12))
65 plot2d(t,f4(t+19))
66 plot2d(t,f5(t))
67 plot2d(t,f6(t+4))
68 plot2d(t,f7(t+12))
69 plot2d(t,f8(t+19))
70 title("courbes des moteurs pour le bat monkey")
71 xlabel("t [~ms]")
72 ylabel("angle(t) [rad]")
73
```

Voici à quoi ressemble l'interface de l'application :

On peut y contrôler la vitesse d'escalade du BatMonkey (entre 1 et 10, avec un rapport de 3 entre le minimum et le maximum), la mettre en pause, voir les angles des moteurs, ou l'étape actuelle d'escalade(entre 0 et 255). La connexion n'est pas encore fonctionnelle car, de par des problèmes d'horloge, la trame reçue est, au choix, une partie de la trame complète, plusieurs répétitions de la chaîne, ou même rien si les délais ne sont pas respectés (ce qui arrive trop souvent malheureusement).



Pendant cette séance, nous avons monté le corps du BatMonkey pour pouvoir y attacher les membres. Nous avons dû utiliser des équerres pour pallier le manque de solidité de notre boîte, qui se désagrgeait sous le poids des moteurs. Voici une photo de l'apparence actuelle du BatMonkey :

La carte Arduino est dans le dos du BatMonkey, seuls ses bras sont correctement positionnés ici.

Ilane a préparé une tête pour notre singe sur Onshape, nous l'imprimerons bientôt pour l'intégrer au robot.

Nous allons aussi devoir régler les problèmes d'angles qui persistent malgré toutes nos précautions, et trouver un moyen de retrouver la trame complète dans les émissions Bluetooth pendant la semaine et la prochaine séance.

