Avancement du stage

Modification mécanique

Démontage des blocs balais principaux, balais latéral et aspiration

- Etant donné que nous n'utilisons pas la partie nettoyage du robot je l'ai enlevé pour gagner de l'espace pour une batterie supplémentaire et pour gagner en poids. Il n'y a pas eu besoins d'utiliser de leurre pour que le robot continu de fonctionner normalement sans les trois moteurs (moteur balais principale, moteur balais latéral, moteur d'aspiration) et sans le capteur de saleté (situé dans le bloc balais principale).

Photo du Roomba sans les blocs de nettoyage:

Observation:

- Dans l'état actuel (sans batterie supplémentaire) le robot est trop léger pour que les roues motrices soit suffisamment enfoncé dans leurs blocs respectif (un ressort permet de faire en sorte que les roues touche toujours par terre, mais il est trop "dur" pour le poids du robot sans la partie nettoyage) ce qui empêche le robot d'avancer normalement. Solution temporaire: mise en place d'un poids à l'arrière du robot.

Modification de l'alimentation

Estimation de la consommation et de l'autonomie des différents éléments

- Le robot Roomba avec le système de nettoyage consomme environ 26 watts, la Kinect consomme environ 13 watts et un noeud FIT consomme au maximum 7 watts. Pour l'instant la solution retenue serait d'alimenter les trois éléments cités ci dessus à l'aide d'une batterie de 14,4 Volts et une capacité de 6000mA/h (voir plus). Ainsi on obtiendrait une autonomie d'environ 1h40.
- Concernant l'ordinateur portable, il utiliserait sa propre batterie qui lui confère une autonomie d'environ 4h.

Etude du rechargement des batteries

- Le rechargement de la batterie du Roomba ce fera avec le dock de la même manière que pour l'utilisation normale du Roomba. Avec le dock et la batterie d'origine (14,4 V, 3000 mAh) le Roomba se recharge avec un courant de 1,13 Ampères à une tension allant de 15,5 à 18 Volts.

- L'ordinateur sera alimenté par sa propre batterie qu'il faudra recharger. Nous ne savons pas encore quelle solutions nous allons choisir pour recharger l'ordinateur (utilisation d'un deuxième dock, utilisation du dock d'origine...) mais pour pouvoir évaluer le courant qu'il faut pour recharger l'ordinateur j'ai fait plusieurs tests en alimentant l'ordinateur à l'aide d'une batterie 12 Volts et d'un adaptateur allume-cigare :
  - Ordinateur sans batterie, processeurs utilisés à 50% : environ 1,5 Ampères.
  - Ordinateur avec la petite batterie chargée, processeurs utilisés à 50% : environ 1,5 Ampères.
  - Ordinateur avec la grosse batterie chargée, processeurs utilisés à 50% : environ 1,5 Ampères.
  - Ordinateur avec les deux batteries chargées, processeurs utilisés à 50% : environ 1,5 Ampères.
  - **Conclusion :** lorsque les batterie son chargées (ou enlevées) l'ordinateur consomme environ 1,5 Ampères sur l'alimentation externe (avec l'adaptateur allume-cigare).
  - Ordinateur avec la petite batterie en charge, processeur utilisés à 50% : environ 3,7 Ampères.
  - Ordinateur avec la grosse batterie en charge, processeur utilisés à 50% : environ 3,8 Ampères.
  - Ordinateur avec les deux batteries en charges, processeur utilisés à 50% : environ 3,8 Ampères.
  - Consommation maximale : deux batteries en charges, processeur utilisé à 100% : environ 4 Ampères.
  - **Conclusion :** il au moins pouvoir débiter 4 Ampères pour pouvoir recharger l'ordinateur.

Dans l'hypothèse ou l'ordinateur serait rechargé à l'aide du dock, il faudrait que le dock puisse débiter un courant totale d'environ 7 Ampères (batterie Roomba plus ordinateur). J'ai démonté le dock pour voir si les composants à l'intérieur pouvait supporter ce courant. La partie puissance est controlée via un transistor MOSFET ZXMP6A17. Dans la documentation on peut voir qu'il peut supporter environ 3 Ampères mais il n'y a pas de dissipateur thermique monté sur le composant (le composant risque donc de ne pas tenir les 3 Ampères à cause de la chaleur). Il faudrait donc remplacer ce composant par un composant semblable mais supportant 8 Ampères.

Photos du dock démonté et du composant à changer :

J'ai trouver sur internet un composant du même constructeur qui pourrait peut être convenir : ZXMP6A16K (à utiliser avec un dissipateur

thermique).

Etude des différentes solutions concernant l'alimentation

1. Création d'un dock pour l'ordinateur au dessus du dock du Roomba. Le dock du Roomba permettrait de charger la batterie du Roomba et du noeud FIT, la batterie serait du même type que celle d'origine du Roomba mais avec une plus grande capacité (6000 mAh). Le dock de l'ordinateur permettrait de charger l'ordinateur et une batterie 12 Volts pour alimenter la Kinect. Travail à faire :
    - Conception de la partie mécanique du dock de l'ordinateur (connecteurs). Utilisation d'une alimentation d'ordinateur classique. Réalisation d'un régulation 12 Volts pour recharger la batterie de la Kinect.
    - On peut éventuellement "booster" le dock du Roomba pour obtenir un rechargement plus rapide de la batterie du Roomba. Pour cela il faut rajouter un dissipateur thermique sur le composants ZXMP6A17 pour qu'il puisse supporter 3 Ampères. On peut utiliser une alimentation d'ordinateur classique en 19,5 Volts pour alimenter le dock du Roomba avec un courant plus important (test fait avec une alimentation 19,5 V, 6,7 A : le Roomba ce met bien en charge et la balise marche correctement).
2. Utilisation du dock d'origine pour recharger à la fois le Roomba et l'ordinateur. Le dock d'origine du Roomba permettrait de recharger la batterie du Roomba, du noeud FIT, de la Kinect (ou utilisation d'une deuxième batterie propre à la Kinect) et de recharger l'ordinateur. Travail à faire :
    - Il faut "booster" le dock du Roomba pour qu'il puisse débiter au moins 6 Ampères. Pour cela il faut remplacer le composant ZXMP6A17 par un composant qui admet un plus fort courant : ZXMP6A16K (à utiliser avec un dissipateur thermique). On peut utiliser une alimentation d'ordinateur classique en 19,5 Volts pour alimenter le dock du Roomba avec un courant plus important.
    - Il faut modifier la partie commande du dock du Roomba pour qu'il ce met en charge non plus avec l'impédance du Roomba mais avec celle du Roomba plus l'ordinateur.
    - Il faut vérifier que la répartition du courant de charge entre la batterie du Roomba et l'ordinateur soit cohérente (qu'il n'y est pas que l'un ou l'autre qui se charge).
3. Utilisation de l'alimentation d'origine du dock du Roomba pour alimenter la balise et utilisation d'un autre alimentation et du connecteur du dock pour recharger le Roomba et l'ordinateur en même temps. Cette solution consiste à garder la balise du dock alimenter par l'alimentation d'origine et ne pas utiliser la partie électronique du dock pour le chargement (détection de l'impédance...). Le chargement s'effectue à l'aide du connecteur d'origine présent dans le dock sur lequel le roomba vient se brancher. Les deux bornes de ce connecteur sont reliées à une alimentation de 20 Volts pouvant délivrer environ 7 Ampères. Travail à faire :
    - Il faut modifier le dock pour ajouter l'alimentation de 20 V, 7 A servant à la charge.
    - Il faut vérifier que la répartition du courant de charge entre la batterie du Roomba et l'ordinateur soit cohérente (qu'il n'y est pas que l'un ou l'autre qui se charge).
    - Il faut réfléchir sur le fait que lorsque le Roomba se recharge normalement su le dock, la balise envoie un signal pour dire au éventuels autres Roombas que le dock est déjà occupé. Il faudrait trouver un moyen d'obtenir le même résultat.
    - Il faut trouver un moyen pour protéger le connecteur d'un éventuel court-circuit.

En attendant de choisir une de ces solutions on peut déjà commander certains composants :

- Thermistance : 10KF3975DPGS (thermistance équivalente à celle présente dans la batterie d'origine du Roomba). Lien chez Radiospare vers une thermistance qui semble être équivalente : NTC THERMISTOR 10K 3%.
- 12 batteries NiMH? 1.2 Volts, 6000 mAh, taille D de chez RadioSpare? : Accumulateur D NiMH, 1,2V 6000mAh ou 12 batteries NiMH? 1.2 Volts, 8000 mAh, taille D de chez Conrad : Accu à cosses U D (R20) NiMH 1,2V 8000 mAh Microbatt.
- Transistor MOSFET : ZXMP6A16K (pour remplacer le composants du dock). Lien chez RadioSpare? : MOSFET P-ch 8,2 A 60 V 0,085R 9,7 W DPAK.

Test de l'autonomie et du temps de rechargement du Roomba

- Pour avoir une idée précise de l'autonomie du Roomba sans la partie aspiration nous avons fait un test avec la batterie d'origine du Roomba (14,4 Volts, 3000 mAh) et un poid à l'arrière. Nous avons lancé le Roomba en mode CLEAN jusqu'à ce qu'il n'est plus de batterie, en le relançant à chaque fois qu'il avait fini son cycle de nettoyage. Nous avons fait le test dand un couloir rectiligne avec de la moquette. Le Roomba à fonctionné pendant 4h45 avant qu'une lumière orange signifiant une batterie faible ne s'allume. Après le test la batterie avait une tension de 11,7 Volts et le Roomba a mit 2h30 pour ce recharger complètement.
- Ensuite nous avons rajouté une batterie 14,4 Volts, 3000 mAh en parallèle avec la batterie d'origine pour augmenter l'autonomie. Nous avons refait le même test que précédemment mais en rajoutant un poids de 2,5 Kilos pour reproduire le poids du noeud FIT, le poids de l'ordinateur et le poids de la Kinect. Nous avons obtenu une autonomie d'environ 5h. Après le test les batteries avaient une tension de 13 Volts.

Le Roomba avec l'équivalent d'une batterie de 14,4 V, 6000 mAh et un poids correspondant à l'ordinateur, la Kinect, et le noeud FIT consomme donc environ 18 Watts. Si on connecte la Kinect et le noeud FIT sur la batterie du Roomba on obtient un total de 38 Watts ce qui nous donne une autonomie d'environ 2h15.

Solution choisie

Nous avons décidé de réaliser la première solution (utilisation du dock d'origine pour recharger à la fois le Roomba et l'ordinateur) :

- Le dock de l'ordinateur à été réaliser avec une prise téléphonique femelle dans laquelle vient ce brancher une plaque recouverte de cuivre des deux côté, attaché su le robot. Ce système à l'air de plutôt bien marcher.
- Nous avons remplacé la batterie d'origine du Roomba par 12 batteries NiMH? 1.2 Volts, 6000 mAh de tailles D. Nous avons réutilisé la thermistance de la batterie d'origine que nous avons placé à côté des nouvelles batteries.

Programmation du Roomba

L'utilisation du Roomba ce fait via une liaison série. On peut envoyer des commande au Roomba (déplacement...) et recevoir des informations sur l'état des différents capteurs du Roomba. Pour communiquer avec le Roomba nous avons choisi d'utiliser le langage Python pour sa simplicité et sa rapidité à être mit en oeuvre. Le constructeur propose une documentation (ROI : Roomba Open Interface) référençant toute les commandes pouvant être envoyé au Roomba. Il existait déjà une bibliothèque Python utilisant ces commandes pour contrôler le Roomba Create (Roomba vendu exclusivement pour la robotique, sans partie aspiration) mais la série Roomba 500 possède des fonctionnalités supplémentaire que la série Create n'a pas (et certaine fonctionnalités existe seulement sur la série Create). J'ai donc créer une nouvelle bibliothèque spécialement pour la série Roomba 500 en m'inspirant de celle déjà faite. A l'aide de cette bibliothèque il est très facile de contrôler le Roomba.

Caractérisation du Roomba

1. **Odométrie :** le Roomba peut renvoyer directement la distance qu'il a parcouru et son angle de rotation depuis la dernière demande effectué. Après quelques tests je me suis rendu compte que ces deux informations étaient très peut précises. Le Roomba peut également renvoyé le nombre de points codeur de chaque roues. J'ai testé cette fonctionnalité et j'ai trouvé que la résolution des codeurs était de 512 points par tour de roue, ce qui n'est pas trop mauvais. J'ai donc fait un programme pour trouver la position du robot directement via le nombre de points codeur de chaque roues. Ainsi on obtient une position beaucoup plus précise. En ligne droite on peut avoir une position assez précise, avec moins de 1% d'erreur. En revanche lorsque le Roomba tourne on perd beaucoup en précision. J'ai réalisé un test
2. ou le Roomba part du dock (position x=0mm, y=0mm, z=0°) et réalise un parcourt de quelques mètres (fig 0.1) A la fin de son parcourt le Roomba retourne au dock et affiche une position : x=106mm, y=-577mm, z=-3°. J'ai refait ce test avec un parcourt beaucoup plus long (plusieurs dizaines de mètres) (fig 0.2)

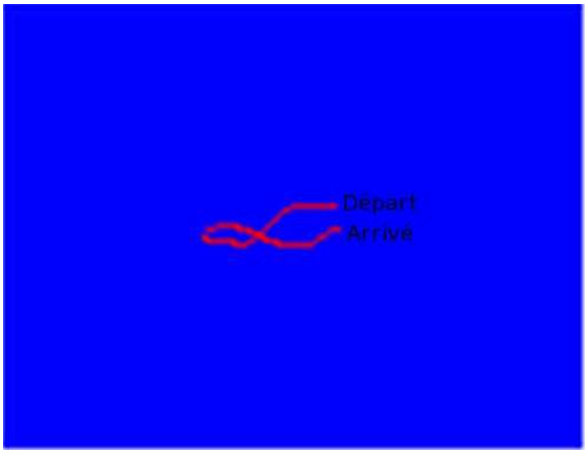

Figure 0.1 : Parcours court

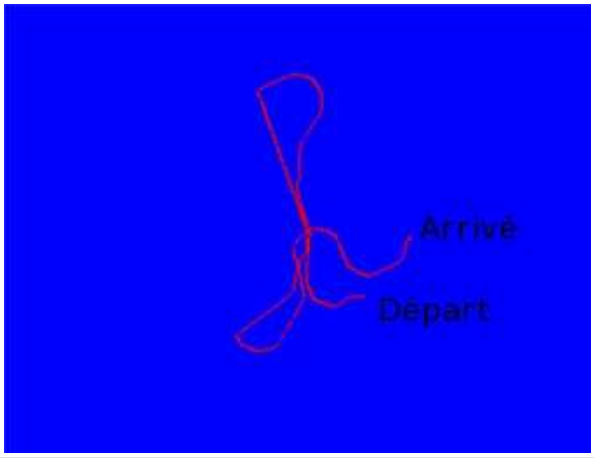

Figure 0.2 : Parcours long

1. A la fin de son parcourt le Roomba retourne au dock et affiche une position : x=2350mm, y=3254mm, z=436°. La position par odomètrie est donc peut précise.
2. **Capteurs de proximités :**
   - Capteurs infrarouges du bumper : il y a 6 capteurs infrarouges réparties dans le bumper du Roomba. Il permettent de détecter un mur ou tout autre obstacle ce trouvant devant ou sur les côtés du robot. Ces capteurs sont des capteurs rectiligne, ils renvoient une valeur en fonction de la puissance du rayon réfléchie. Plus un objet est proche plus la puissance du rayon réfléchie sera grande. Pour une même distance cette valeur peut varier en fonction de la couleur de l'objet réfléchie. Le roomba renvoie deux variables pour chaque capteurs, une variable qui est l'image de la puissance du rayon réfléchie et une autre qui prend comme valeur 1 ou 0 en fonction de la proximité d'un objet. J'ai regroupé dans un tableau la distance à partir de laquelle cette dernière variable passe à 1 pour un objet de couleur blanche :

	Left	Front left	Center left	Center right	Front right	Right
Distance (cm)	10	15	15	15	15	10
Force du signal	300	125	125	125	125	300

   - Capteurs infrarouges de détection de marche ou de trou : il y a 4 capteurs infrarouges réparties sous le Roomba vers l'avant qui permettent de détecter une marche ou un trou qui ce présente devant le Roomba. De la même manière que les capteurs du bumper ils renvoient deux variable, une pour la force du signal et une pour l'état (présence de trou : 1, ou détection du sol : 0). J'ai regroupé dans un tableau la distance à partir de laquelle le robot ne détecte plus le sol (passage à 1 de la variable état) pour un objet de couleur blanche :

	Left	Front left	Front right	Right
Distance (cm)	8	8	8	8
Force du signal	80	80	80	80

1. **Balises :**
   - Mur virtuel : le mur virtuel permet d'empêcher le Roomba d'aller à certains endroits en créant un mur que le Roomba ne pourra

traverser. Il y a deux moyen de tester la présence d'un mur virtuel, soit en regardant la variable du mur virtuel (1 : mur virtuel, 0 : pas de mur virtuel), soit en regardant le code reçu par le récepteur infrarouge omnidirectionnel (code 162 : mur virtuel). Le mur virtuel est détectable jusqu'à 150 cm.
    - Dock : le dock dispose de trois émetteurs infrarouges. Un qui permet de savoir si le Roomba est proche du dock (champ de force), et deux autres qui permettent de déterminer si le Roomba est plus à gauche ou à droite du dock (bouée rouge et verte). Le champ de force a une porté de 60 cm et les bouées ont une portées de 270cm. Le dock envoie un code au Roomba pour lui permettre de ce localiser, d'éviter le dock, ou de vérifier si le dock est libre ou non.
2. **Comportement à deux robots :**
    - Lorsqu'un Roomba occupe un dock et qu'un deuxième Roomba en mode Dock passe à côté le second Roomba ne remarque pas le dock. En effet lorsqu'il est occupé le dock ne renvoie pas de signal. Le deuxième robot va donc éviter le premier Roomba et le dock comme s'il s'agissait d'un vulgaire obstacle.
    - Lorsque deux Roomba se rencontre il ne communique pas entre eux. Ils se contentent de s'éviter comme de vulgaire obstacle.

Réparation du Roomba

Lors d'un test d'autonomie un des Roomba eu un problème au niveaux de la roue gauche. La carte de puissance du moteur gauche fut cassé. Nous supposons que cela est peut être due au fait que nous avions mis trop de poids sur le Roomba pour simuler le poids de l'ordinateur, de la Kinect, du noeud FIT... Pour tenter de réparer le Roomba nous avons commandé des diodes 1N5819, des transistors KSB772YSTU? et des transistors 2SD882. Ainsi nous avons de quoi réparer un hacheur quatre cadrans complet. La référence des transistors PNP et NPN que nous avons commandé n'est pas exactement la même que celle des transistors d'origine de la carte de puissance mais ce sont des transistors équivalent. Une fois les composants reçus nous avons cherché quels composants ne marchait plus sur la carte mère du Roomba et nous les avons remplacé. Il a fallu remplacer en tout une diode, un transistor NPN et un transistor PNP. La réparation fut un succès le Roomba remarche normalement.

Scénario 1

Le scénario 1 consiste à lancer le mode clean à la réservation du Roomba et lancer le mode dock lorsque le Roomba n'est plus réservé. Le périmètre accessible au Roomba est délimité par une bande blanche au sol. Pour simuler ce scénario sans noeud FIT (donc sans réservation) le Roomba démarre un cycle de nettoyage de 10 minutes, cherche le dock et se recharge pendant 15 minutes. Il réalise cela un nombre de fois déterminé à l'avance :

- Après quelques essais nous nous somme rendus compte que le connecteur mini-din 7 n'était pas très fiable. En effet, apparemment après plusieurs cycles à cause des vibrations et des choques le connecteur ce déconnecte légèrement ce qui provoque des erreurs dans la lectures du port série. Il faudra donc remplacer ce connecteur par un autre plus approprié pour une application finale à long terme. Après avoir remplacé le connecteur mini-din 7 des erreurs subsistent dans la réception sur le port série. Ces erreurs doivent donc venir du programme mais étant donné qu'elles apparaissent de manière aléatoire j'ai du mal à les corriger. Je peut néanmoins utiliser un programme plus simple qui fonctionne mieux, il ne calcul pas la position du Roomba mais cela n'est pas utile pour le moment pour le scénario 1.
- A l'aide de ce programme simplifié nous avons lancé le Roomba dans le cycle suivant : 10 minutes en mode Clean, 30 minutes de recharges sur le dock, et tout cela 25 fois. Le Roomba à effectué ce cycle de 17h15 jusqu'à 10h15 le lendemain (17h au total). Le fichier log retraçant les différentes étapes du cycle est disponible ici. Le Roomba a effectué le cycle comme prévue sans dépasser les lignes blanche. Le résultat de cette expérience est plutôt concluant. L'expérience étant mené dans le couloir derrière le Hall Robotique, le Roomba a mis en moyenne 2 minutes pour trouver le dock à la fin du mode Clean.

Après avoir testé la robustesse de ce premier scénario nous avons rajouter des fonctionnalité pour surveiller la température de la batterie du Roomba, vérifier le niveau de charge de la batterie de l'ordinateur (pour ne pas qu'il s'éteigne pendant le cycle) et avoir une position odométrique (même si elle n'est pas précise) :

- Le premier test fut un échec, le Roomba fut retrouvé en dehors de la zone délimitée par les lignes blanches. Apparemment l'ordinateur se serait éteint pendant le cycle (la vérification du niveau de charge ne fonctionne donc pas à cause d'une erreur dans le programme) et le Roomba serait resté en mode Clean ou Dock sans se soucier des lignes blanches.
- Pour tester la durée de vie des batteries du Roomba et de l'ordinateur nous avons fait un second test : 10 minutes en mode Clean, 20 minutes de recharges sur le dock, et tout cela 8 fois. Cela a pris 3h, le fichier log retraçant les différentes étapes du cycle est disponible ici. Il en résultat la courbe suivante :

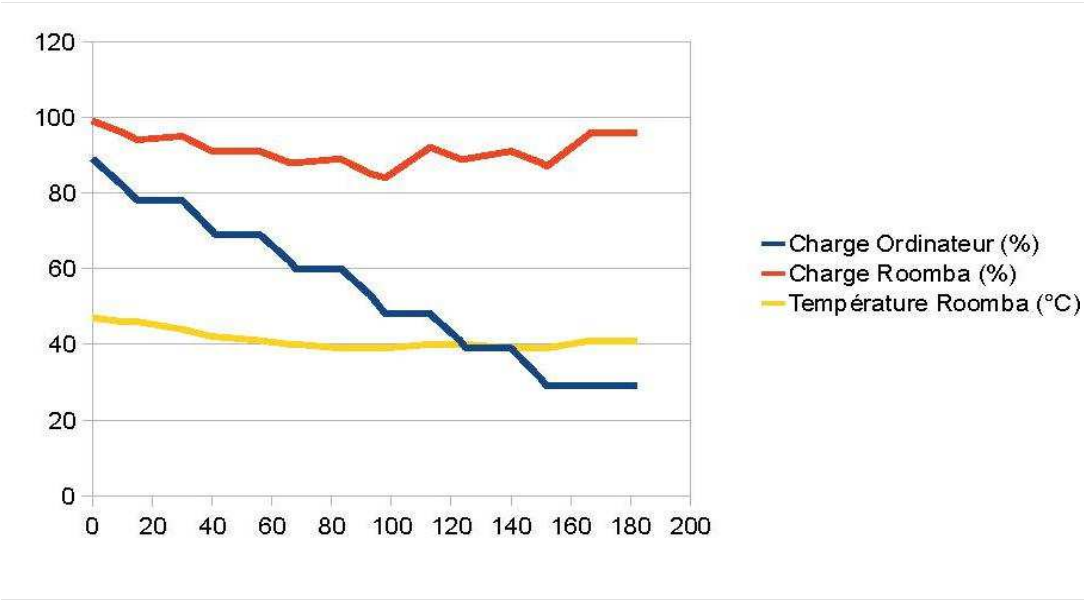

Figure 1

- Comme on peut le voir sur la courbe la batterie de l'ordinateur ne se recharge en fait jamais. Cela est du au fait que pour que les batteries de l'ordinateur se recharge il faut une tension de 19,6 Volts et un troisième fil. Sur le connecteur que nous avions fait il n'y avais que la tension de 19,6 Volts, pas de troisième fil. L'ordinateur reconnaissait qu'il était branché mais ne se chargeait pas, il évitait juste de se décharger. Nous avons donc rajouter un petit connecteur pour ce troisième fil et maintenant l'ordinateur ce charge correctement. Une fois le connecteur modifié nous avons refait la même manipulation (dont le fichier log ce trouve ici) qui a donnée les résultats suivant :

Temps (min)	Charge ordinateur (%)	Charge Roomba (%)	Température Roomba (°C)
0	110	96	26
10	93	93	27
11	93	92	27
31	99	100	31
41	91	97	32
42	90	96	32
62	98	97	32
72	91	94	33
75	89	93	33
95	98	100	39
105	90	96	39
120	79	92	40
140	94	96	39
150	87	93	40
152	85	93	39
172	97	100	41
182	89	97	43
186	86	96	43
206	97	96	42
216	89	93	41
217	88	92	41
237	98	92	40
247	90	89	39
248	89	89	39

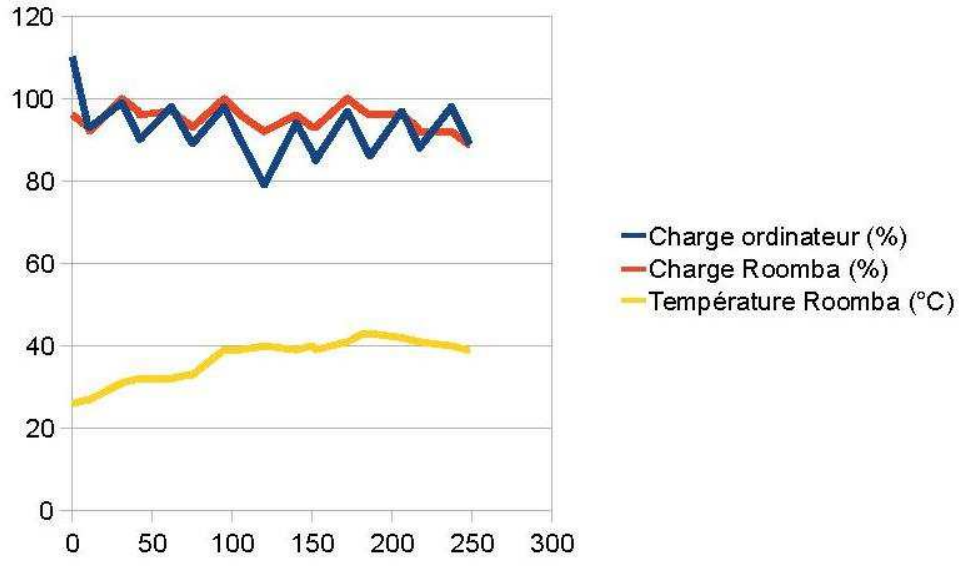

Figure 2

- L'ordinateur peut donc maintenant bien se recharger. Nous avons ensuite fait un dernier test sur les batteries. Cette fois nous avons lancé le mode Clean jusqu'à ce que les batteries de l'ordinateur ou du Roomba soient chargés à moins de 30%, après cela le Roomba passe en mode Dock et ce recharge jusqu'à ce que les batterie de l'ordinateur et du Roomba soient au dessus de 90%. Nous avons fait deux fois ce cycle, au départ la batterie de l'ordinateur était à moitié pleine et celle du Roomba était pleine. Le fichier log retraçant les différentes étapes du cycle est disponible ici. L'expérimentation a donné les résultats suivant : graphique 3.1 et graphique 3.2. L'ordinateur ce décharge donc beaucoup plus vite que le Roomba, ce sera lui qui limitera la durée d'un cycle.

Temps (min)	Charge ordinateur (%)	Charge Roomba (%)	Température Roomba (°C)
0	68	99	45
47	29	85	43

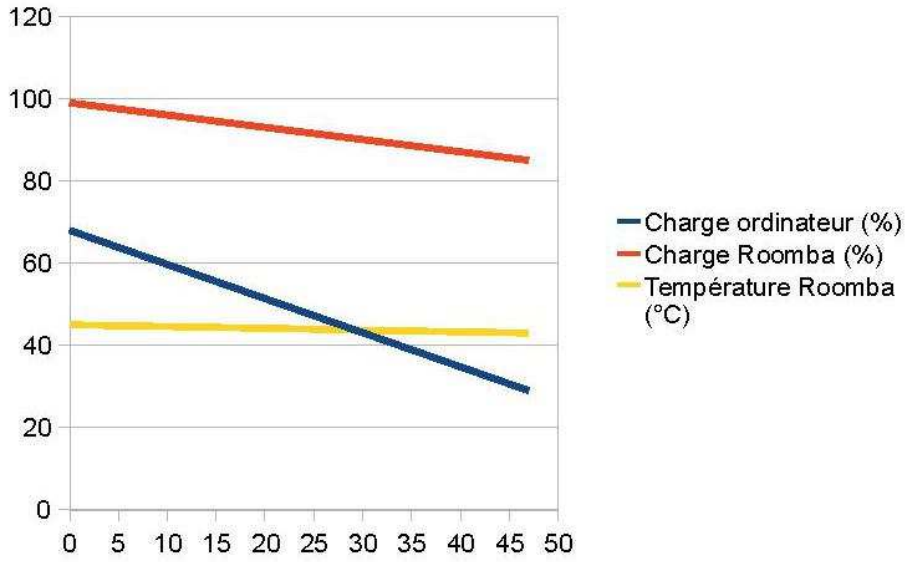

Figure 3.1

Temps (min)	Charge ordinate
0	
72	

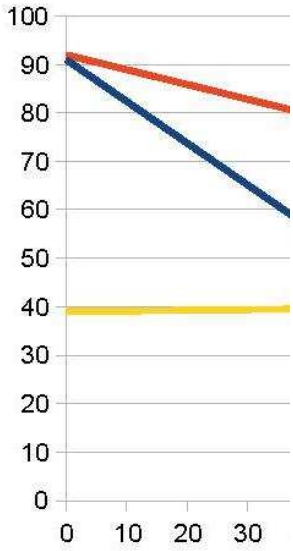

Figure 3.2

- Nous avons ensuite testé ce scénario dans le couloir regroupant les bureaux du SED. Nous avons placé une bande blanche devant chaque bureau pour empêcher le Roomba d'y pénétrer mais cela rallonge le temps que met le Roomba pour se docker. Le Roomba met en moyenne 4 à 5 minutes pour se docker :

Trial number	Time for docking (s)
1	146
2	321
3	273
4	203
5	110
6	444
7	600
8	92
9	169
10	76
11	659

Valeur moyenne :
281,1818181818

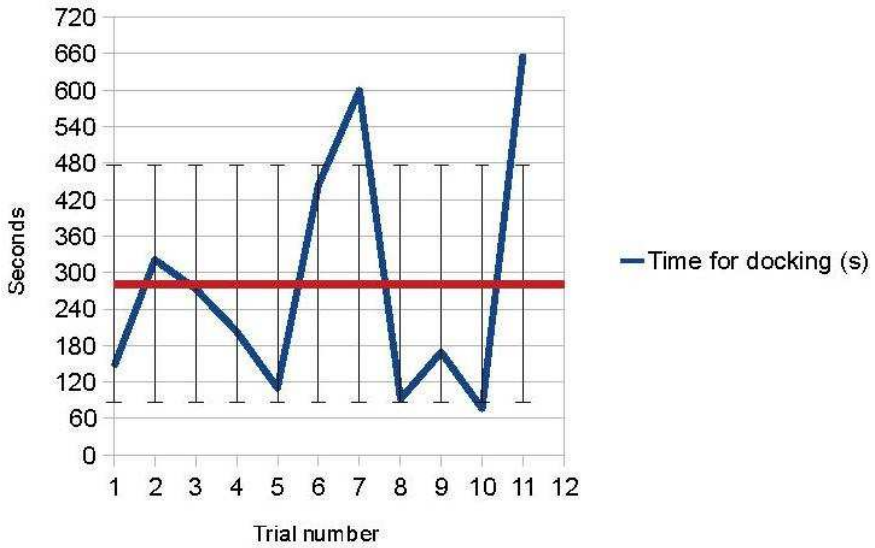

Figure 4

Vidéo de démonstration ici.

Vidéo montrant en détail les mouvements du Roomba (accéléré deux fois) ici.

Scénario 2

Le scénario 2 consiste à réaliser des trajectoires simples (ligne droite, tourner sur lui même) pour avoir une position odométrique plus précise. Un cycle est relativement court et ce termine par un dockage ce qui permet de "remettre à zéro l'odométrie" pour que les erreursde l'odométrie de s'additionne pas au court du temps. Pendant un cycle le Roomba recule pour sortir du Dock, tourne d'un quart de tour, avance jusqu'à ce qu'il rencontre un obstacle ou une ligne blanche, fait demi-tour et revient se docker. Après quelques tests nous nous sommes rendu compte que le dock ne permet pas de remettre à zéro l'odomètrie. En effet lorsque le Roomba se dock il peut prendre un angle différent a chaque fois, il ne partira donc pas droit et risque de rencontrer un mur qui le bloquera au retour.

Scénario 3

Le scénario 3 consiste à réaliser un Roomba suiveur de ligne. Le Roomba sortira du Dock, suivra une ligne (d'une largeur de 7,75 à 8 cm) fera demi-tour et suivra cette même ligne pour retourner au Dock. La position odométrique devrait être relativement précise pour ce scénario.

- Les premiers essais sont plutôt concluant, le Roomba suit bien la ligne. On peut ainsi lui faire faire plusieurs allers-retours avant de le faire se docker. Néanmoins il ne peut pas prendre de virage.
- j'ai ensuite essayé de modifier le programme d'asservissement pour qu'il puisse prendre des virages mais je ne suis pas parvenu à avoir le résultat souhaité, soit le Roomba oscillait en ligne droite et prenait correctement les virages soit il avançait bien droit en ligne droite mais ne pouvait pas prendre les virages. Finalement nous avons opté pour une autre solution, à chaque virage ou fin de ligne le Roomba fait un tour sur lui même pour repérer si la ligne continue et vers ou elle continue. Ainsi l'asservissement permet au Roomba d'aller droit et il peut prendre un virage de 90° ou plus. J'ai commencé à tester ce scénario sur le long terme en faisant 4 fois 10 aller-retour le long d'une ligne blanche avec un virage, ce scénario semble très robuste.

Vidéo de démonstration ici.

Programmes Python

Voici l'ensembles des programmes python que j'ai conçus :

- Le module servant de base à tout mes programmes (à importer à chaque fois donc) c'est une version du module fourni pour le Roomba Create que j'ai modifié pour l'utilisation que nous voulions en faire : roomba.py
- L'interface graphique permettant de contrôler le Roomba, faite avec Tkinter (de nombreuse fonctionnalités ne sont pas intégrées, pour l'instant il est seulement possible d'utiliser les modes de base du Roomba, de le déplacer, de suivre sa position et voir sa trajectoire) : roombaInterface.py
- Le scénario 1 (pour voir en détail en quoi consiste ce scénario consultez le paragraphe correspondant) : scenario1.py
- Le scénario 2 (pour voir en détail en quoi consiste ce scénario consultez le paragraphe correspondant) : scenario2.py
- Le scénario 3 (pour voir en détail en quoi consiste ce scénario consultez le paragraphe correspondant) : scenario3.py

Pour envoyer des commandes et recevoir des informations du Roomba il faut importer le module roomba.py. Dans ce module il y a la classe Roomba500 qui permet d'ouvrir une connexion avec le Roomba. Cette classe dispose de toutes les méthodes nécessaire pour utiliser le Roomba.

Si vous souhaitez seulement recevoir des informations du Roomba de temps en temps vous pouvez utiliser les méthodes de la classe telles qu'elles. En revanche si vous souhaitez recevoir les informations du Roomba de manière continu (pour une position odométrique à l'aide des informations codeurs par exemple) il existe une méthode (la dernière de ce module) qui, une fois lancé en tâche de fond avec Multiprocessing, permet de contrôler et recevoir toute les information du Roomba. Les calculs de l'odométrie ce font directement dans cette méthode. Pour envoyer une commande et recevoir les informations il est nécessaire de dialoguer avec cette méthode lancé en tâche de fond à l'aide de deux Queue (files d'attentes). Ces deux Queue sont défini dans le constructeur de la classe il est donc inutile d'en définir d'autre. C'est en utilisant cette méthode que j'ai réalisé mais programmes. Voici un petit exemple d'utilisation :

```
robot=roomba.Roomba500(serialportname) # création de l'objet robot à l'aide de la classe Roomba500 du module roomba, serialportname est le nom (ou le numéro sous windows) du port sur lequel est connecté le Roomba

p = multiprocessing.Process(target = robot.interfaceSerial) # création de la méthode interfaceSerial en tâche de fond

p.start() # début du processus

robot.qsend.put({'changeMode': 'clean'}) # utilisation de Queue pour envoyer la commande au processus interfaceSerial, la commande est d'appeler la méthode changeMode de la classe Roomba500 avec l'argument 'clean' : le Roomba passe en mode Clean

sensor_list =robot.q.get() # récupération des informations du Roomba avec Queue, cette commande renvoie un liste contenant l'état de tout les capteurs et la position du Roomba

print sensor_list[roomba.POS_X] # impréssion de la position x du Roomba, la variable POS_X du module roomba permet de facilement trouver une information spécifique de la liste complète de toutes les information sensor_list

p.terminate() # termine le processus interfaceSerial
```

```
robot.close() # méthode close de la classe Roomba500, ferme la liaison série et éteint le Roomba
```

-- LeoStatkus - 31 Aug 2012

I	Attachment	Action	Size	Date	Who	Comment
	Photo.jpg	manage	131.5 K	21 Jun 2012 - 14:20	LeoStatkus	
	Photo_du_21-06-12_à_15.58.jpg	manage	54.2 K	21 Jun 2012 - 14:09	LeoStatkus	
	ROOMBA_detection2ligne.mp4	manage	42506.8 K	29 Aug 2012 - 09:45	LeoStatkus	
	ROOMBA_suivi2ligne.mp4	manage	32759.8 K	29 Aug 2012 - 09:45	LeoStatkus	
	demontage_bloc_nettoyage_avec_batterie.jpg	manage	50.4 K	20 Jun 2012 - 12:07	LeoStatkus	
	gr.jpg	manage	71.8 K	13 Sep 2012 - 17:37	RogerPissard	
	graph.jpg	manage	39.3 K	13 Sep 2012 - 19:36	RogerPissard	
	graph2.jpg	manage	133.2 K	13 Sep 2012 - 19:11	RogerPissard	
	graph3_(glissé(e)s).jpg	manage	52.9 K	13 Sep 2012 - 19:21	RogerPissard	
	graph3_(glissé(e)s)_1.jpg	manage	55.2 K	13 Sep 2012 - 19:25	RogerPissard	
	log.txt	manage	9.1 K	24 Jul 2012 - 12:55	LeoStatkus	
	log2.txt	manage	4.9 K	25 Jul 2012 - 13:32	LeoStatkus	
	log3.txt	manage	5.9 K	26 Jul 2012 - 13:24	LeoStatkus	
	log4.txt	manage	4.2 K	27 Jul 2012 - 14:07	LeoStatkus	
	map_complique.jpg	manage	15.9 K	11 Jul 2012 - 09:56	LeoStatkus	
	map_simple.jpg	manage	16.5 K	11 Jul 2012 - 09:30	LeoStatkus	
	roomba.py.txt	manage	22.7 K	30 Aug 2012 - 12:43	LeoStatkus	
	roombaInterface.py.txt	manage	12.3 K	30 Aug 2012 - 12:43	LeoStatkus	
	scenario1.py.txt	manage	12.1 K	31 Aug 2012 - 09:41	LeoStatkus	
	scenario2.py.txt	manage	7.9 K	30 Aug 2012 - 12:43	LeoStatkus	
	scenario3.py.txt	manage	11.9 K	30 Aug 2012 - 12:43	LeoStatkus	
	video_roomba.mp4	manage	18092.1 K	31 Aug 2012 - 12:38	LeoStatkus	

--- This topic: SED/Suivi > WebHome > SuiviStagiaires2012 > SuiviLeoStatkus > AvancementLeoStatkus
Topic revision: 13 Sep 2012, RogerPissard