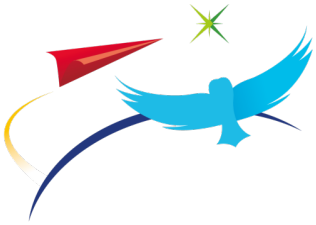


Institut Supérieur de l'Aéronautique et de l'Espace

IS 202 - Avant-Projet Spatial

Conception d'un robot d'exploration

Enrico DE DONATI – Simone URBANO – Tao LIU
David PORTILLA ABELLAN - Vincent SCHAEFFER

13/03/2014

Introduction

L'objectif de ce projet est de concevoir un robot LEGO, qui représente un rover martien d'exploration et qui répond à un cahier des charges.

La mission principale est de recueillir des échantillons et de les apporter au site de récupération. Cela consiste en l'exploration d'une zone d'intervention limitée, afin de localiser et de ramener des échantillons. Avant de se déplacer, le robot se tient en position d'attente au point O du plan du terrain (Figure 1), orienté dans la direction de l'axe y. Quand le robot reçoit l'autorisation, il commence la mission. Une fois que l'échantillon est installé sur la zone d'exploration, le robot détecte sa position, s'approche vers l'échantillon, l'apporte dans la zone de récupération et retourne dans la zone d'attente. Enfin, le robot s'arrête automatiquement.

Plus précisément, pour détecter la position de l'échantillon, un capteur à ultrason est utilisé. En utilisant le capteur RGB, le robot peut détecter la frontière du terrain. Un capteur tactile est utilisé pour réaliser les actions de prise et de dépose de l'échantillon. Le robot est piloté par trois moteurs. Deux des trois moteurs contrôlent le déplacement et la rotation du robot. L'autre contrôle le bras du robot. Toutes les parties du robot sont traitées et contrôlées par un microprocesseur en utilisant la programmation JAVA.

La réalisation du projet commence par une conception fonctionnelle basée sur la démarche d'Ingénierie Système. Ensuite, en utilisant les pièces fournies dans le kit de LEGO, le robot de mission est construit. Le contrôleur NXT, qui contient un microprocesseur, est utilisé pour contrôler le robot, lequel est piloté à partir de l'outil de programmation JAVA. Après avoir réalisé les tests, le robot est amélioré et optimisé pour qu'il soit rapide et robuste.

Les travaux sont repartis en cinq parties correspondant aux parties du rapport :

1. Analyse fonctionnelle du système
2. Conception et construction du robot
3. Stratégie suivie
4. Ecriture du code
5. Tests de validation

Les parties 1 et 2 ont été réalisées de manière linéaire. En revanche, les parties 3, 4 et 5 ont été réalisées de manière itérative. Notre stratégie initiale a évolué en fonctions des possibilités du code, et des résultats des tests.

Le terrain

Le terrain d'intervention est un plan de couleur blanc et uniforme, mesurant 1500mm x 2500mm (Figure 1). Les quatre côtes du terrain sont limités par des bandes noires qui font 50mm de largeur. La zone d'attente est un carré noir mesurant 500 mm x 500 mm. La zone de récupération est composée d'un bac rouge, de 200 de diamètre et de 40mm de hauteur. Son épaisseur est comprise entre 1 et 2mm. Elle est fixée sur le terrain. Sa position est donnée sur la figure 1. L'échantillon est posé sur un support qui est un cylindre noir de diamètre 20mm et de hauteur 36mm. La position du support est donnée par hasard sur le terrain.

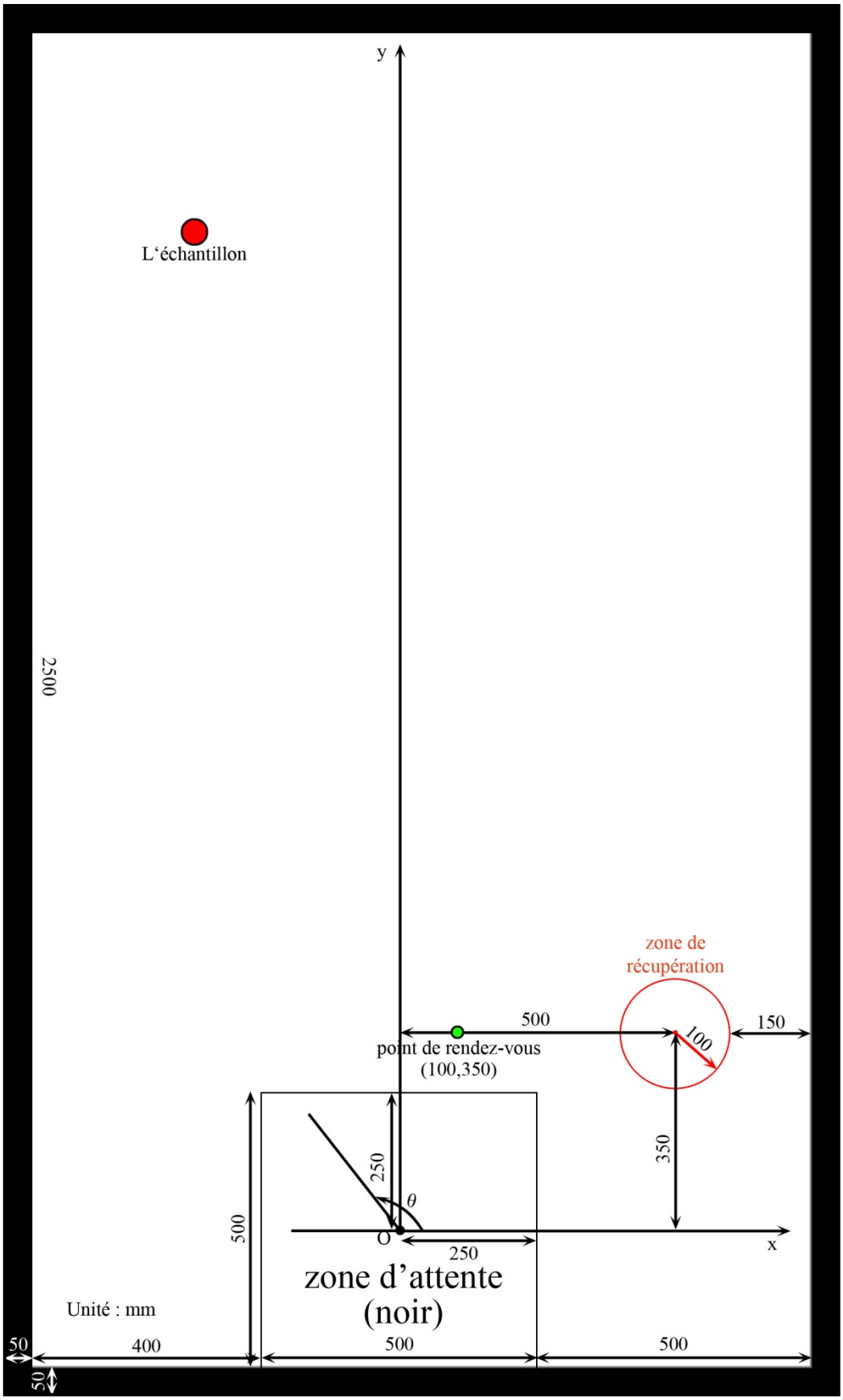


Figure 1 – Plan du terrain
Orientation de l'angle theta / emplacement de l'échantillon

1 Analyse fonctionnelle

1.1 Cycle de vie

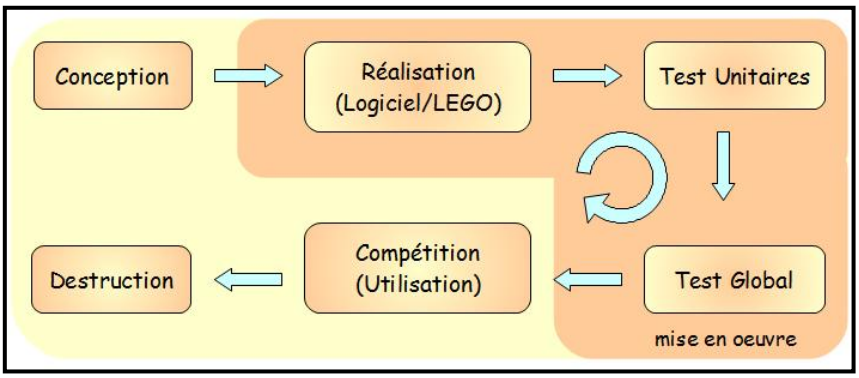
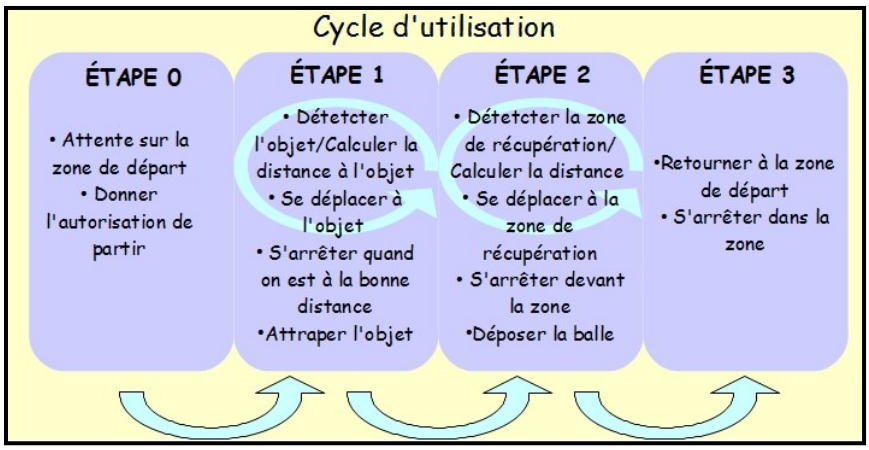


Figure 2 - Cycle de vie du robot

1.2 Cycle d'utilisation et cas d'utilisation



Cas d'utilisation

- La boule est placée au centre de la zone.
- La boule est proche d'un bord.
- La boule est devant la zone de récupération.
- La boule est derrière la zone de récupération.
- La boule est très proche de la zone de départ

Figure 3 - Cycle d'utilisation et cas d'utilisation

1.3 Modèle entités-relations (ER)

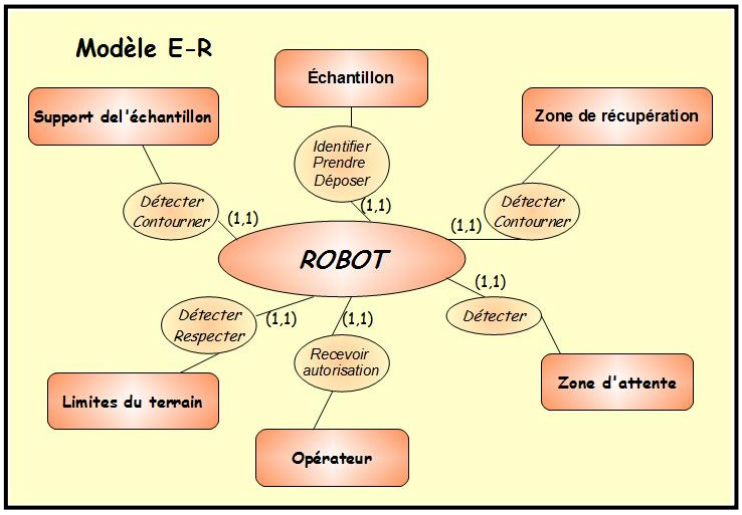


Figure 4 - Modèle entités-relations

1.4 Fonctions principales et fonctions contraintes

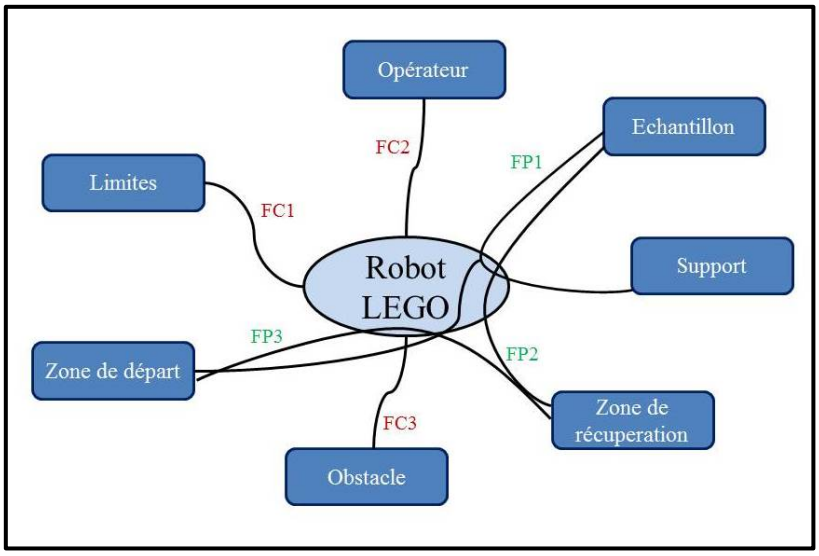


Figure 5 – Diagramme pieuvre

FP1 : Récupérer l'échantillon sur le support
- **FP1.1 :** Détecter l'échantillon
  - **FP1.1.1 :** Démarrer le capteur ultrason
  - **FP1.1.2 :** Explorer la carte
  - **FP1.1.3 :** Déterminer l'angle et la distance à l'échantillon
- **FP1.2 :** Se rapprocher de l'échantillon
  - **FP1.2.1 :** Se déplacer vers l'échantillon
  - **FP1.2.2 :** Actualiser la trajectoire
  - **FP1.2.3 :** Ralentir avant d'arriver
- **FP1.3 :** Attraper l'échantillon
  - **FP1.3.1 :** Déployer le bras
  - **FP1.3.3 :** Récupérer l'échantillon

FP2 : Déposer l'échantillon dans la zone de récupération
- **FP2.1 :** Détecter la zone de récupération
  - **FP2.1.1 :** Explorer la carte
  - **FP2.1.2 :** Déterminer l'angle et la distance à la zone de récupération
- **FP2.2 :** Se rapprocher de la zone de récupération
  - **FP2.2.1 :** Se déplacer vers la zone de récupération
  - **FP2.2.2 :** Actualiser la position
  - **FP2.2.3 :** Ralentir avant d'arriver
- **FP2.3 :** Déposer l'échantillon
  - **FP2.3.1 :** Déployer le bras
  - **FP2.3.2 :** Lâcher l'échantillon dans la zone de récupération
  - **FP2.3.3 :** Ranger le bras en position de repos

FP3 : Retourner à la zone de récupération
- **FP3.1 :** Se déplacer vers la zone de départ
- **FP3.2 :** Arrêter la mission dans la zone de départ

FC1 : Respecter les limites du terrain
FC2 : Attendre l'autorisation à l'opération
FC3 : Eviter les obstacles (zone de récupération et support de l'échantillon)

1.5 Enhanced Functional Flow Block Diagram (eFFBD)

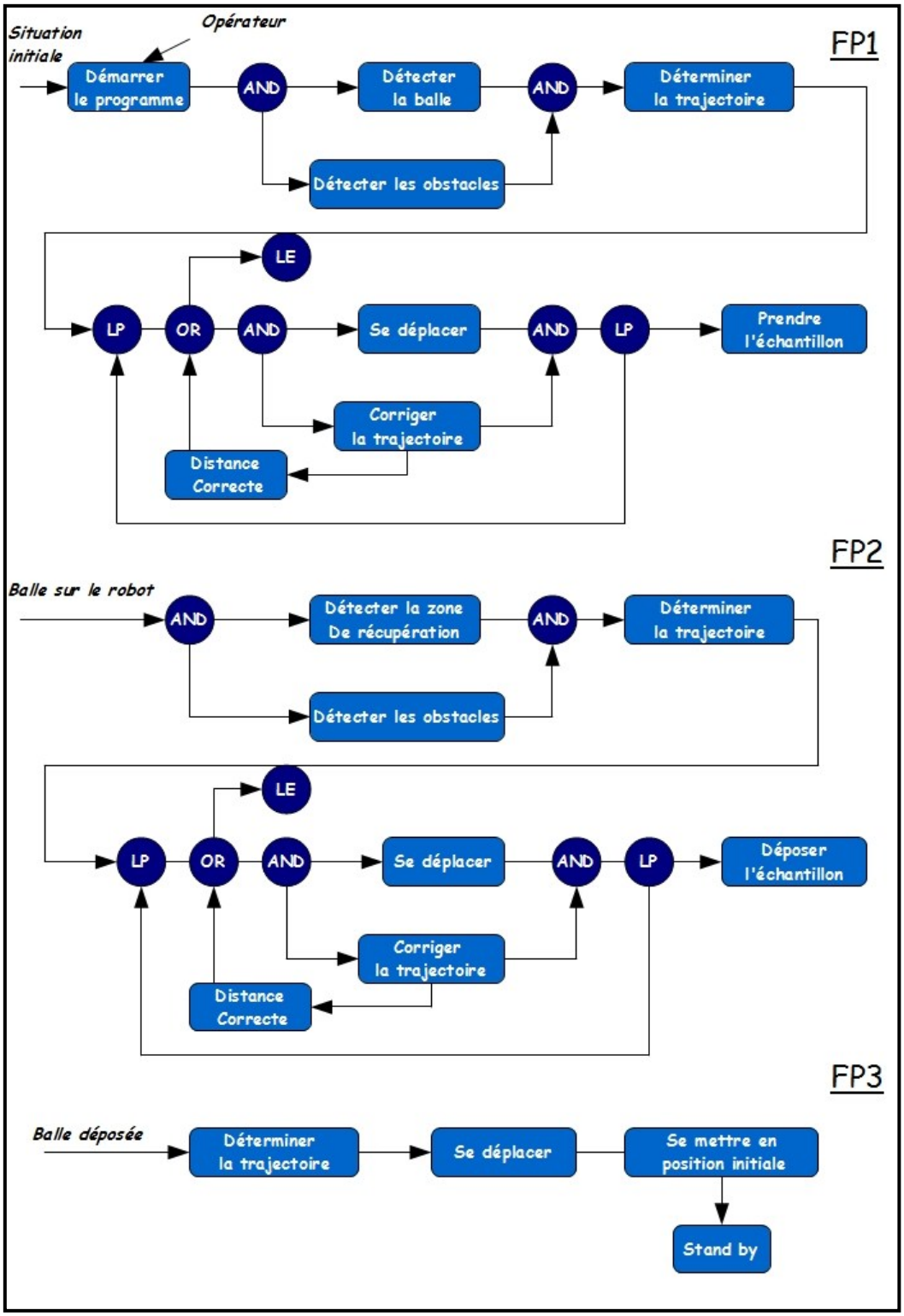


Figure 6 – eFFBP pour les fonctions principales

1.6 Physical Bloc Diagram (PBD)

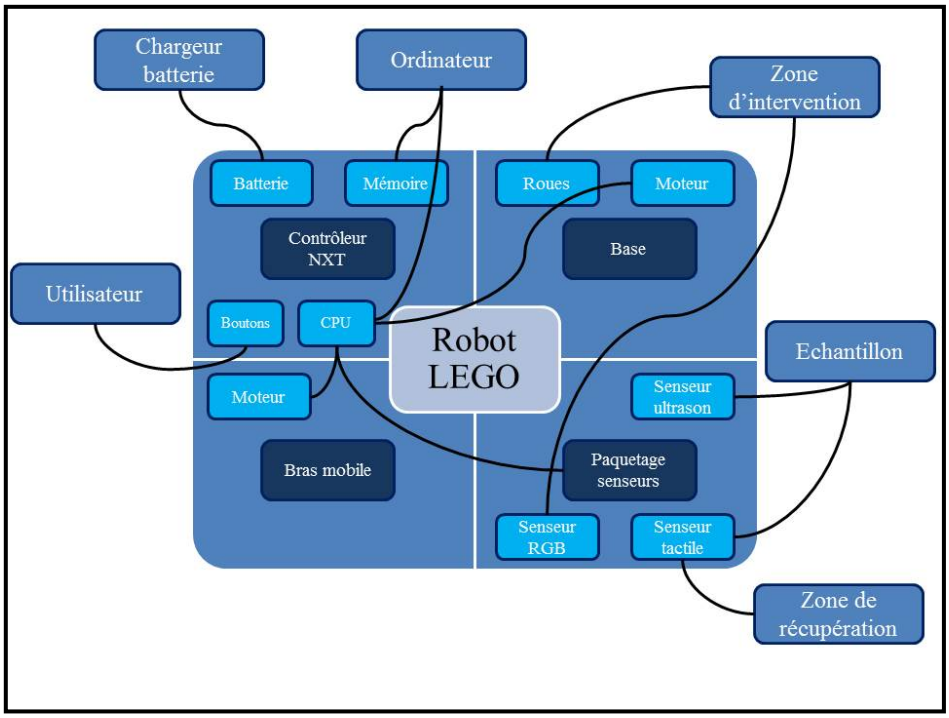


Figure 7 – Physical Bloc Diagram

1.7 Exigences et contraintes

Comprendre le contexte dans lequel le robot travail se traduit par : trouver le but principal de la mission, les objectifs secondaires et les exigences/contraintes du robot.

Le tableau suivant présente de manière schématique le contexte opérationnel du robot.

Contexte	
Mission	Rechercher un échantillon et le rapporter à la zone de récupération
Finalité	Récupérer un echantillon

Exigences	**Contraintes**
Etre rapide (moins de 5 min)	Etre robuste
Capable de se déplacer sur l'ensemble de la zone	Détécter le terrain
Détécter l'objet à récupérer	Rester dans les limites du terrain
Capable d'attraper un objet	Etre autonome
Détécter la zone de récupération	Éviter les "obstacles"
Déposer l'échantillon dans le bac de récupération	
Détécter la zone de départ	
Connaitre sa position et son orientation	

Tableau 1 – Exigences et contraintes du robot

2 Conception du robot

2.1 Composition du robot

Le robot est construit avec les pièces fournies par le kit LEGO. En particulier, parmi toutes les pièces, on dispose de :

- Un contrôleur NXT
- Trois moteurs électriques
- Un senseur ultrason
- Un senseur RGB
- Deux senseurs tactiles

Chacune de ces pièces possède des contraintes particulières par rapport à la fonction qui lui est assignée suite à l'analyse fonctionnelle. La composition du robot a été guidée par la nécessité de respecter ces contraintes, mais également dans le but d'obtenir une architecture finale solide.

Nous commencerons par une description générale de l'architecture finale, puis nous analyserons chaque partie en détails.

2.2 Architecture générale

On peut décomposer le robot en quatre partie: la **base**, le **bras**, le **contrôleur NXT** et le **paquetage senseurs**.

La **base** doit être très solide parce qu'elle doit supporter le poids de toute la structure. La base est composée essentiellement des deux moteurs avec lesquels on contrôle la translation et la rotation du robot. Elle est liée avec le contrôleur NXT et le paquetage senseurs. Le **paquetage senseurs** est positionné à l'avant du robot et est composé des quatre senseurs disponibles. Au dessus de la base, nous avons attaché le **contrôleur NXT**, ce choix permet un accès facile aux boutons et au port USB nécessaire au chargement du logiciel. Enfin, le **bras** a pour fonction d'attraper et de transporter la balle. Il est commandé par le troisième moteur et est placé à l'avant, dessus du paquetage senseurs. En position de repos, le bras est plié à l'arrière et s'appuie sur une structure, ce qui permet au contrôleur NXT de rester accessible.

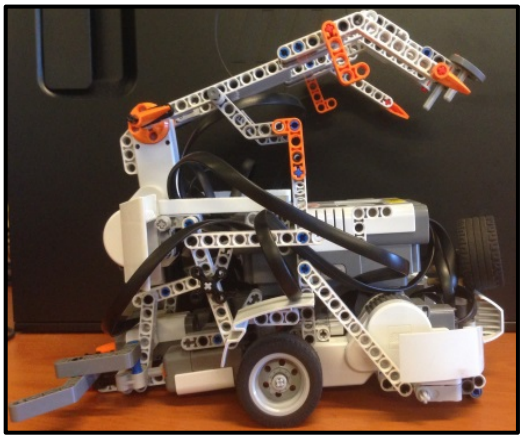


Figure 8 – Vue latérale (bras en position de repos)

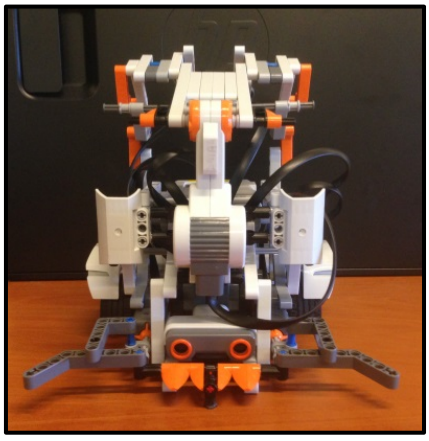


Figure 9 – Vue frontale

2.3 Base et moteurs : déplacement et rotation

Nous avons choisi d'utiliser deux roues motrices. Pour assurer le déplacement du robot ainsi que sa rotation, le deux roues motrices son liées à deux moteurs différents. Pour la translation, les moteurs tournent dans le même sens, mais pour obtenir une rotation les moteurs tournent dans des directions opposées. Pour garantir le bon fonctionnement de la rotation on a minimisé le frottement de l'appui à l'arrière en concentrant le poids du robot sur les roues motrices.

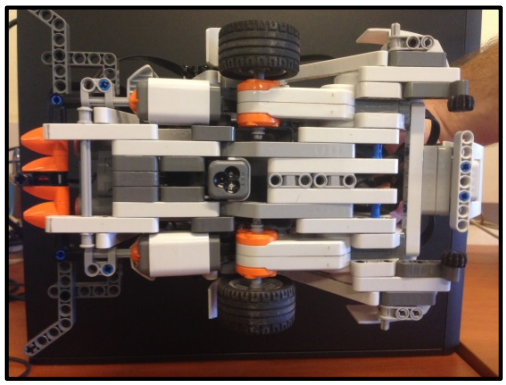


Figure 10 – Vue de la base

2.4 Paquetage senseurs : navigation

Le paquetage senseurs était la partie la plus difficile à concevoir et à construire à cause des nombreuses contraintes à respecter. Dans la mesure où c'est la partie nécessaire à la navigation, on a choisi de poser les quatre senseurs à l'avant du robot.

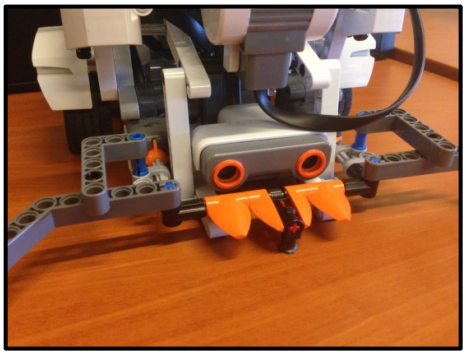


Figure 11 – Senseur ultrason (cercles oranges) et barre liant les deux senseurs tactiles (pics oranges)

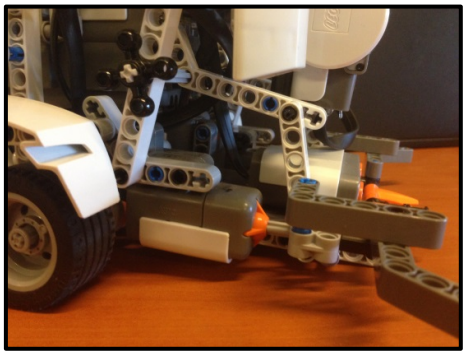


Figure 12 – Senseur tactile (vue latérale)

Le placement du **senseur ultrason** a été le plus complexe, car il doit être à l'avant et ne subir aucune obstruction qui puisse compromettre son fonctionnement. De plus, on doit tenir compte du fait qu'il doit être lié au contrôleur NXT par un câble. Il est utilisé pour « explorer » la zone d'intervention et détecter l'échantillon et la zone de récupération. Il a été plus facile de placer le **senseur RGB**, qui sert à détecter les limites de la zone d'intervention. Pour cette raison, il pointe vers le bas et est placé derrière le senseur ultrason. La seule contrainte du senseur RGB est le rayon d'action de 5mm. Enfin, les **senseurs tactiles** n'ont pas de contraintes particulières. Il est possible de leur attacher des barres LEGO pour les « allonger ». On a choisi de les liés en créant un barre unique positionné juste à l'avant du senseur ultrason, en faisant attention de ne pas obstruer le champ de ce dernier. Les senseurs tactiles servent à détecter la présence du support de l'échantillon et de la structure de la zone de récupération.

2.5 Bras : collecter l'échantillon

Le **bras mobile** a pour fonction d'attraper et de transporter l'échantillon. En position de repos, il est plié à l'arrière pour ne pas créer pas d'obstruction au senseur ultrason. Pour attraper l'échantillon le bras tourne grâce au troisième moteur. Avec le bras dans cette position, la vision du senseur ultrason est obstruée. La navigation est ainsi commandée par les senseurs tactiles pendant cette phase. Après avoir ramené l'échantillon, le bras tourne vers le haut en position de transport, et quand les senseurs signalent la présence de la zone de récupération, le bras s'abaisse et dépose l'échantillon. Lorsque le bras est en position de transport, cela ajoute du poids à l'avant. Pour équilibrer le robot, nous avons rajouté des pièces à l'arrière n'ayant aucune fonction structurelle.

Figure 13 – Vue latérale avec le bras déployé

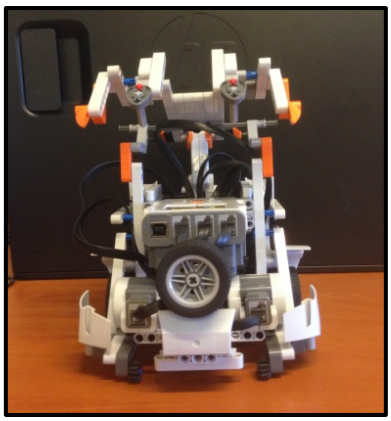


Figure 14 – Vue arrière

2.6 Contrôleur NXT : commander le robot

Le **contrôleur NXT** permet de charger le logiciel et de commander les moteurs et les senseurs. Il faut que les boutons et que le port USB le chargement de logiciel soient accessibles. Pendant la construction, nous avons dû prévoir le fait que le contrôleur NXT doit être lié par des câbles aux moteurs (au nombre de 3) et aux senseurs (au nombre de 4).

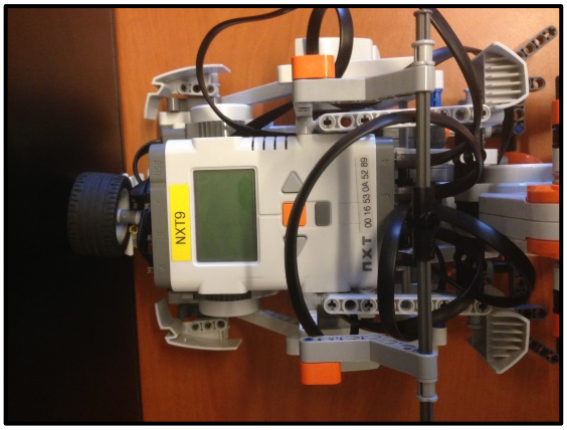


Figure 15 – Contrôleur NXT

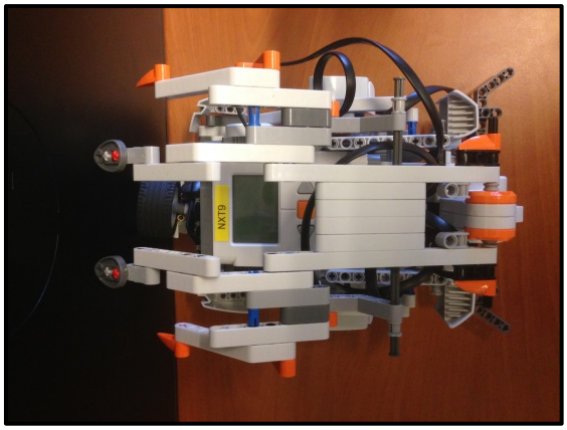


Figure 16 – Bras en position repos

3 La stratégie

Dans ce paragraphe, nous allons décrire avec précision la stratégie de notre robot. Le système de coordonnées choisies (x, y, θ) est représenté sur la figure 1. Le robot est localisé par la position du milieu de ses deux roues, et par son orientation. D'autre part, le robot, par l'intermédiaire de la classe LeJOS OdometryPoseProvider, actualise automatiquement sa position.

3.1 Détection de la balle

La distance maximale de détection du senseur ultrason n'étant pas précise (environ 600mm), nous avons réfléchi à deux stratégies différentes pour détecter l'échantillon.

Option 1 : (Figure 17) Le senseur ultrason est utilisé pour détecter la balle. La première stratégie contient quatre points de détection dont les coordonnées sont (0,0), (0, 600), (0,1200) et (0,1800). Les points de détection sont choisis par rapport à la distance maximale détectable par le senseur ultrason. Au départ, le robot se situe au centre de la zone d'attente (0,0). Le robot est orienté dans la direction de l'axe y. La détection commence par θ = 210° et se termine à θ = -30°. Pour ce faire, le robot tourne d'abord de 120°. Ensuite il tourne de 240° jusqu'à l'orientation -30° dans le sens horaire en même temps qu'il recherche l'échantillon. S'il trouve la balle, il corrige sa position, et s'en approche. S'il ne la trouve pas, le robot se réaligne dans la direction de l'axe y et avance de 600mm jusqu'au point (0,600). Il recommence ensuite la même opération que précédemment. S'il ne détecte de nouveau pas la balle, il avance encore de 600mm dans la direction de l'axe y jusqu'au point (0,1200) afin de refaire une détection.

Option 2 : (Figure 18) Considérant que le senseur ultrason n'a pas une précision parfaite à une distance supérieure à 600mm, la deuxième stratégie est mise au point comme une solution plus robuste. Celle-ci contient 8 points de détection. Ils sont aux coordonnées (-150,0), (250,0), (250,600), (-150,600), (-150,1200), (250,1200), (250,1800) et (-150,1800). A l'état initial, le robot se situe au point (-150, 0) du plan dans la zone d'attente. Le robot est orienté dans la direction de l'axe y. La méthode de détection est similaire à celle de la première stratégie sauf que la ligne de parcours du robot est différente. S'il détecte la balle, il s'arrête et corrige sa position et puis il s'en approche. Sinon, il avance de 400 mm dans la direction de l'axe x jusqu'au point (250,0). Il refait à nouveau une détection dans le sens antihoraire. S'il ne trouve de nouveau pas la balle, il avance jusqu'au point (250,600), et il refait la détection dans le sens antihoraire .Pour détecter tout le terrain, il reste seulement à parcourir tout les points de détection dans l'ordre (250,600), (-150,600), (-150,1200), (250,1200), (250,1800), (-150,1800).

Remarque : Le capteur ultrason ne fait pas la différence entre le support de l'échantillon et la zone de récupération. Pour ne pas confondre les deux, nous déterminons les coordonnées du point détecté (calculées à partir de la position du robot, de l'orientation du robot, et de la distance données par le capteur ultrason). Si ces coordonnées se trouvent à moins de 150 mm du centre de la zone de récupération, nous savons que le capteur ultrason a détecté la zone de récupération. Dans le cas contraire, nous avons détecté le support de l'échantillon.

3.2 Prise de la balle

Dès que le robot trouve la balle, il s'arrête. Il se rapproche à une distance de 300mm du support et affine son orientation. Pour ce faire, il tourne d'abord de 90°. Puis, il tourne vers la droite jusqu'à ce qu'il détecte le support. Il enregistre son orientation (angle1). Il tourne à nouveau vers la droite jusqu'à ce que le capteur ultrason ne détecte plus rien. Il enregistre à nouveau son orientation (angle2). Ces deux angles sont différents car le capteur ultrason possède une ouverture angulaire d'environ 60°. Le robot tourne ensuite d'un angle de k*(angle1-angle2)/2 pour s'aligner parfaitement face au support. Le facteur k vient du fait que l'ouverture angulaire du capteur ultrason n'est pas

centrée avec l'axe du capteur. Ensuite, il abaisse son bras, avance de 300mm, puis relève le bras. En fin il recule pour s'éloigner du support.

3.3 Apport de la balle à la zone de récupération

Pour apporter la balle à la zone de récupération, le robot se rend à un point de rendez-vous situé aux coordonnées (100,350). Il s'oriente dans la direction θ = 0°, face à la zone de récupération. Il avance ensuite et s'arrête dès que les senseurs tactiles détectent la zone, et abaisse enfin son bras pour déposer la balle.

3.4 Retour à la zone d'attente

Après avoir déposé la balle, le robot replace son bras dans la position de repos et recule au point de rendez-vous. Enfin, il retourne à la zone d'attente et s'arrête de façon autonome.

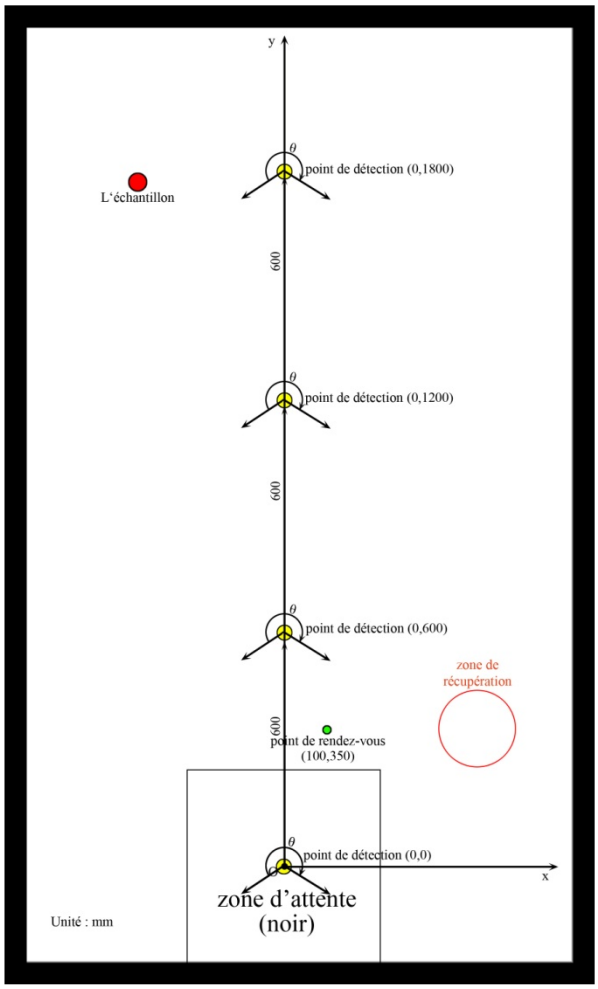


Figure 17 – Stratégie 1

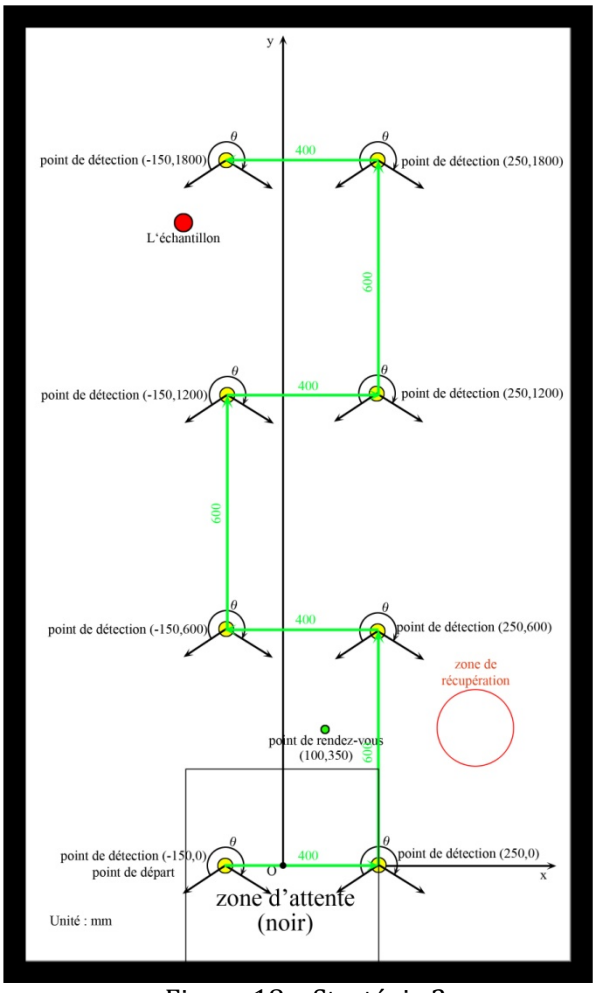


Figure 18 – Stratégie 2

4 Logiciel

4.1 LEGO Mindstorms NXT

4.1.1 Présentation de l'architecture

Le kit NXT vient avec un contrôleur programmable, aussi appelé Brique intelligente. Cette Brique (voir figure 18 pour son diagramme de blocs) présente un processeur principal 32-bit ARM avec 64 kB de RAM et 256 kB de mémoire Flash qui fonctionne à 48 MHz. Pour seconder le processeur principal 8-bit, le coprocesseur AVR est joint. Le processeur principal et le co-processor communiquent périodiquement à travers un I^2C bus.

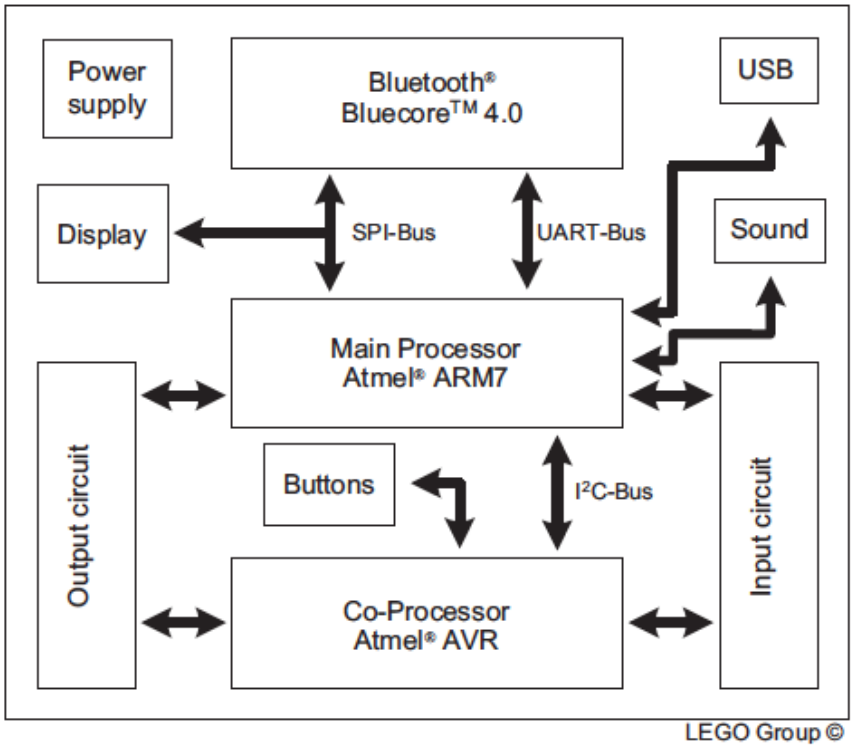


Figure 19 – Le bloc LEGO NXT

Il vient aussi avec trois ports de sortie bidirectionnels, pour connecter et contrôler des actionneurs (mécanismes) comme des moteurs électriques ou des actionneurs linéaires, et quatre ports qui supportent des senseurs analogiques et digitaux.

Les communications avec la Brique sont possibles soit par USB, via un full-speed USB 2.0 port, soit par Bluetooth, disponible à travers un CSR BlueCore 4 chip connecté au USART du ARM.

Le USB 2.0 est normalement utilisé pour le connecter à un PC, et le Bluetooth communiquer avec d'autres Briques NXT ou d'autres dispositifs munis de connexions Bluetooth comme les *Smartphones*, les *Tablettes*, etc.

En haut de la Brique il y a un panneau d'affichage connecté au processeur principal via un SPI bus (*Serial Peripheral Interface Bus*), et quatre boutons en caoutchouc, contrôlés par le co-processor, pour interagir avec la Brique.

La Brique NXT vient aussi avec un amplificateur audio connecté au contrôleur ARM PWM (Pulse-Width Modulation), et un microphone de 16 Ω avec une bande passante de 2 – 16 kHz.

4.1.2 Ports de sortie

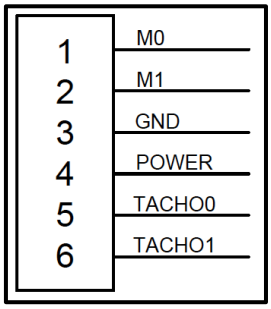


Figure 20 – Schéma général des ports de sortie

Les trois ports de sortie fonctionnent de la même façon. Ils ont une fiche terre (GND) et ils donnent une alimentation électrique de 4,3 V (POWER). Deux signaux (M0 et M1), qui viennent d'un *H-bridge motor driver* interne, contrôlent le moteur en standby, en marche avant, en marche arrière ou en frein. Le driver du moteur est gouverné par les PWM pulses générés par le co-processeur. Il a aussi deux signaux (TACHO0 et TACHO1) connectés au PIO (*Parallel Input/Output controller*) du processeur principal en utilisant un Schmitt trigger pour supprimer le bruit.

4.1.3 Ports d'entrée

Les ports d'entrée se comportent de forme différente selon le type de senseur connecté. Ces ports permettent des interfaces analogiques et digitales.

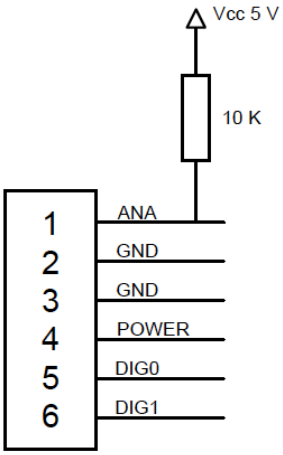


Figure 21 – Schéma général des ports d'entrée

LEGO envisage trois types de senseurs :

- **Des senseurs actifs :** Ces senseurs appartiennent à la version précédente de LEGO MINDSTORM, le RCX. Ils ont besoin d'un câble adapteur NXT.

- **Des senseurs passifs :** Ce sont des senseurs analogiques qui n'ont pas besoin du *special power/measurement* des senseurs actifs. Les besoins de puissance de ces senseurs ne sont pas couvertes via le pin analogique (ANA), mais via un pin spécifique (POWER). A noter que

l'échantillonnage de tous les convertisseurs A/D AVR se produit simultanément, donc les senseurs actifs et passifs doivent être échantillonnés au même taux, 333 Hz. Tous les senseurs dans le LEGO MONDSTORMS NXT sont passifs, sauf le senseur à ultrasons.

- **Des senseurs digitaux :** Ces senseurs ont toutes les ressources pour fonctionner indépendamment. Ainsi, ils accomplissent leur mission de manière autonome et ils envoient et reçoivent des informations au/du ARM via un chanel I2C (DIG0 et DIG1) à 9600 bit/s ou le ARM fonctionne en tant que master. Ces senseurs sont établis comme des aires de mémoire externes où le programmeur peut lire ou écrire pour contrôler le senseur, ou bien récupérer des informations. Le senseur à ultrasons est l'unique senseur digital dans le kit NXT.

4.2 LeJOS NXJ

LeJOS est une petite machine virtuelle de Java apportée à la Brique LEGO NXT. Elle comporte les drivers nécessaires et un ensemble d'APIs pour pouvoir utiliser toutes les prestations du kit NXT d'une façon facile en programmation Java.

4.3 Structure logiciel

Le logiciel développé pour la mission se compose de deux classes : ***NavigatorRobot*** et ***MainClass***.

- ***NavigatorRobot***

En suivant une programmation orientée à objets, cette classe est la représentation de notre robot physique. Il peut rouler, lever et descendre le bras, et utiliser les senseurs.

Pour créer un ***NavigatorRobot***, il faut donner au constructeur les ports physiques où chaque composant se trouve (*left motor, right motor, arm motor, rgb sensor, ultrason sensor, left tactil sensor and right tactil sensor*). On a définie comme attributs le diamètre des roues et l'écart entre elles, qui vont être nécessaire pour utiliser la librairie *DifferentialPilot*. Cette dernière possède des fonctions pour faire rouler le robot tout droit, en arrière, et tourner sur lui même (ce sont principalement celles que l'on utilise).

Lors du mouvement du robot sur la surface connue, on utilise la librairie ***OdometryPoseProvider*** pour connaitre en permanence la position du robot. Cela est nécessaire pour savoir où la zone de récupération et la zone de départ se trouvent par rapport au robot.

Ce robot implémente ***FeatureListener*** ; ainsi quand le senseur à ultrason détecte un obstacle, la méthode ***featureDetected*** sera exécutée. Dans cette méthode on vérifie si l'objet détecté est la zone de récupération. Dans le cas positif, rien ne se passe. Dans le cas négatif, on suppose qu'il s'agit du support de la balle : on enregistre la distance, l'angle, et on met la variable ***objectDetected*** à *true*.

Avec ces idées, on a défini des méthodes pour chaque petite fonction que le robot est capable de faire : se déplacer, tourner, s'aligner vers la balle, prendre la balle, aller à la zone de récupération, s'aligner face à la zone de récupération, et retourner au point de départ.

- ***MainClass***

Dans la méthode *main* qui se trouve dans cette classe, on crée un ***NavigatorRobot*** et on utilise ses méthodes. On a essayé plusieurs stratégies de mission en cherchant fiabilité et vitesse. Par exemple nous réalisions initialement les scans par angle de 240° (le robot finissait le scan même s'il détectait la balle). Puis, nous avons décidé de diviser les scans en 10 petits scans de 30° (Le robot

s'arrête dès qu'il détecte la balle à l'issu d'un petit scan). De cette façon, quand on trouve la balle on est presque aligné, par contre précédemment il fallait tourner à l'inverse.

- **L'algorithme final**

Pour balayer toute la surface, nous devons réaliser quatre scans. Dans une boucle on réalise les scans, et le déplacement vers l'avant façon continue. Quand la balle est trouvée on sorte de la boucle et on appelle les fonctions suivantes (*MyRobot* représente une instance de la classe ***NavigatorRobot***):

```
myRobot.travel(...); //avancer jusqu'à la balle

myRobot.alignToObjectDetected();

myRobot.catchBall();

myRobot.goToRDV(); //avancer jusqu'à la zone de récupération

myRobot.leftBall();

myRobot.backHome();
```

En termes de vitesse on considère que le programme est très optimisé (le deuxième test de la compétition en 52 secondes a été vraiment rapide) ; mais la fiabilité a été quelque fois questionné : le senseur à ultrason n'est pas un laser ; il est donc très difficile de toujours trouver un bon alignement (il arrive même que le robot ne détecte pas la balle). Aussi, il a été difficile de coder avec différents threads : quand un objet est détecté la méthode ***featureDetected*** est exécutée en parallèle avec le main thread, et il faut faire attention par exemple à ne pas bouger les moteurs depuis les deux threads au même temps.

En conclusion de la partie logiciel, nous pouvons dire que nous avons commencé ce module avec une très bonne maîtrise en Java, mais il nous a pris du temps pour nous adapter aux API's de LeJOS. De plus, le code dépend fortement de la forme du robot ; il a donc dans un premier temps été très difficile de faire le code. N'ayant un temps que très limité pour la réalisation de ce projet, l'algorithme que nous avons développé est loin d'être optimisé en robustesse et en vitesse, mais suffisant pour fonctionner dans des cas simples.

5 Plan de test et validation

Afin de valider notre robot, et ses fonctionnalités, nous avons réalisé une série de tests, depuis les mouvements les plus basiques jusqu'à la mission finale. L'organisation des essais a été la suivante :

a) Tests des capteurs
b) Tests des actions de base (avancer, reculer, tourner, bouger le bras)
c) Tests de la localisation (position et orientation)
d) Tests des actions semi-complètes (scan initial, attraper et reposer la boule, revenir au départ)
e) Tests d'adaptation au terrain (bordures, zone de récupération, support de la boule)
f) Tests des actions complètes (FP1, FP2, FP3)
g) Tests des différents cas d'utilisation
h) Tests de rapidité
i) Tests de fiabilité

Dans chaque série de tests, il y a une série de points à valider permettant d'une part de satisfaire les exigences du robot, et d'autre part de nous permettre de réaliser avec sûreté la série de tests suivante. Chaque série de tests présentera les différents objectifs à atteindre, ainsi que les mises en situations permettant vérifier ses objectifs. Finalement, les résultats des tests seront présentés.

a) <u>Tests des capteurs</u>

Objectifs	• Vérifier que les capteurs tactiles fonctionnent. • Vérifier que le capteur ultrason repère les objets. • Déterminer les caractéristiques du capteur ultrason. • Vérifier que le capteur de couleur fonctionne.

Mises en situation / Tâches	Résultats
Faire afficher sur l'écran un 1 lorsqu'un des capteurs tactiles touche le support de la boule.	Réussi
Faire afficher sur l'écran un 1 lorsqu'un des deux capteurs tactiles touche la zone de récupération.	Réussi
Faire afficher la distance repérée par le capteur ultrason sur l'écran en temps réel.	Réussi
Déterminer la distance maximale et minimale que le capteur ultrason peut mesurer.	Distance maximale de détection = 600 mm.
Déterminer la précision du capteur ultrason.	Précision = 2 cm.
Déterminer l'angle d'ouverture du capteur ultrason	Ouverture angulaire de détection de 35° à gauche, et 25° à droite.
Faire afficher un 1 sur l'écran lorsque le capteur de couleur voit du noir.	Réussi

Conclusion	• Ces tests nous ont permis de vérifier le bon fonctionnement des capteurs. • La dissymétrie de l'ouverture angulaire nous a permis de calculer le facteur de correction k nécessaire à l'alignement finale face au support de l'échantillon (voir p11, stratégie, « prise de la balle »). • La distance maximale de détection étant de 600 mm, nous avons opté pour l'option 1 dans la stratégie de « Détection de la balle » (voir p11).

b) Tests des actions de base

Objectifs	• Vérifier que les fonctions avancer et reculer fonctionnent. • Vérifier que les fonctions tourner à gauche et tourner à droite fonctionnent. • Vérifier que les fonctions baisser le bras et lever le bras fonctionnent.

Mises en situation / Tâches	Résultats
Faire avancer le robot sur une distance de 50cm.	A permis de déterminer avec précision le diamètre des roues (= 41,5 mm).
Faire reculer le robot sur une distance de 50cm.	A permis de déterminer avec précision le diamètre des roues (= 41,5 mm).
Faire des corrections sur l'erreur commise entre les deux moteurs.	Légère déviation à gauche, sans conséquences pour notre projet.
Faire avancer le robot sur une durée infinie.	Réussi
Faire reculer le robot sur une durée infinie.	Réussi
Faire tourner le robot à droite de 360°.	A permis de déterminer avec précision la distance entre les deux roues (128 mm).
Faire tourner le robot à gauche de 360°.	A permis de déterminer avec précision la distance entre les deux roues (128 mm).
Faire tourner le robot à droite de 15°.	Vérifié
Faire tourner le robot à gauche de 15°.	Vérifié
Faire baisser le bras de 180°.	Réussi
Faire lever le bras de 180°.	Réussi
Faire baiser le bras de 30°.	Réussi
Faire lever le bras de 30°.	Réussi

Conclusion	• Ces tests nous ont permis de déterminer précisément les caractéristiques géométriques du robot afin d'être le plus précis possible dans la navigation. • Nous avons également déterminé les vitesses de translation et de rotation acceptable pour les déplacements du robot. • Nous avons de même déterminé une vitesse acceptable pour le mouvement du bras.

c) Tests de la localisation

Objectifs	• Vérifier que le robot met à jour sa position à près chaque déplacement. • Vérifier que le robot met à jour son orientation après chaque déplacement.

Mises en situation / Tâches	Résultats
Faire afficher la position (x,y) en temps réel.	Réussi
Faire afficher l'orientation en temps réel.	Réussi

Conclusion	• La bonne localisation du robot nous a permis de vérifier que le robot actualise automatiquement sa position et son orientation de manière précise. • La bonne localisation du robot nous assure une bonne précision pour le retour au point de rendez-vous et le retour à la zone de départ.

d) Tests des actions semi-complètes

Objectifs	• Réaliser un scan initial du terrain pour trouver la boule. • Récupérer la boule sur son support. • Déposer la boule dans la zone de récupération. • Retourner dans la zone de départ.

Mises en situation / Tâches	Résultats
Faire un scan initial avec la boule proche.	Réussi
Faire un scan initial avec la boule éloignée.	Réussi
Faire un scan initial avec la boule très éloignée.	Réussi
Faire un scan initial avec la boule très proche.	Réussi
Faire un scan initial sans la boule.	Réussi
Faire un scan initial avec la boule près du bord gauche.	Réussi
Faire un scan initial avec la boule près du bord droit.	Réussi
Faire un scan initial avec la boule près du bord du fond.	Réussi
Faire un scan initial avec la boule derrière la zone de récupération.	Non réalisé, pas assez de temps pour développer le logiciel afin que le robot fonctionne dans ce cas de figure.
Placer le robot en face du support et récupérer la boule.	Réussi
Placer le robot avec la boule en face de la zone récupération et y déposer la boule.	Réussi
Placer le robot dans un endroit connaissant sa position et son orientation et le faire revenir à la zone de départ.	Réussi

Conclusion	• Grâce à un bon travail préliminaire sur les tests des mouvements de base, de la localisation, et des capteurs, nous avons réussi avec succès les tests sur les actions semi-complètes.

e) Tests d'adaptation au terrain

Objectifs	• Ne pas confondre le support de la boule et la zone de récupération. • S'arrêter si l'on s'approche d'une bordure. • Ne pas confondre la zone de départ er les bordures.

Mises en situation / Tâches	Résultats
Faire un scan initial sans confondre la zone de récupération avec le support de la boule.	Réussi
Faire stopper le robot s'il détecte un obstacle devant lui.	Réussi
Faire avancer le robot en le faisant contourner un obstacle sur son chemin.	Non réalisé, pas assez de temps pour développer le logiciel afin que le robot contourne un obstacle.
Faire stopper le robot s'il détecte une bordure.	Réussi
Faire stopper le robot si ses coordonnées dépassent les limites du terrain.	Non réalisé, pas assez de temps pour développer le logiciel afin que le robot contourne un obstacle.
Faire stopper le robot s'il entre en contact avec un obstacle par les capteurs tactiles.	Réussi
Faire stopper le robot lorsqu'il arrive dans la zone de départ.	Réussi
Faire un parcours où le robot part de la zone départ, va jusqu'à une bordure, s'arrête, fait demi-tour, reviens à la zone de départ, et s'y arrête au centre.	Non réalisé

Conclusion	• Le terrain étant connu à l'avance, nous n'avons finalement que peu d'imprévu dans le parcours. Nous n'avons donc pas dépensé beaucoup de temps pour prendre en comptes les contraintes de la zone d'évolution du robot.

f) Tests des actions complètes

Objectifs	• Tester la fonction principale 1. • Tester la fonction principale 2. • Tester la fonction principale 3.

Mises en situation / Tâches	Résultats
Partir de la zone de départ jusqu'à la récupération de la boule (plusieurs emplacements du support).	Réussi
Partir du terrain avec la boule et la déposer dans la zone de récupération (plusieurs points de départ).	Réussi
Partir de la zone de récupération et revenir à la zone de départ (plusieurs points de départ).	Réussi

Conclusion	• Le bon fonctionnement des actions semi-complètes nous a assuré la bonne réalisation des actions complètes

g) Tests des différents cas d'utilisation

Objectifs	• Tester l'ensemble du programme dans plusieurs situations.

Mises en situation / Tâches	Résultats
Test complet avec échantillon proche.	Réussi
Test complet avec échantillon éloigné.	Réussi
Test complet avec échantillon très éloigné.	Réussi
Test complet avec échantillon très proche.	Réussi
Test complet sans échantillon.	Non réalisé, pas assez de temps pour développer le logiciel afin que le robot contourne un obstacle.

Conclusion	• Une fois les tests sur les actions complètes réalisés, les tests sur les différents cas d'utilisation se sont très bien passés.

h) Tests de rapidité

Objectifs	• Atteindre la vitesse maximale possible.

Mises en situation / Tâches	Résultats
Test complet à vitesse lente.	Réussi
Déterminer la vitesse maximale des déplacements.	Vérifié
Déterminer la vitesse maximale du bras.	Vérifié
Test complet à vitesse maximale.	Réussi

Conclusion	• Une fois que le robot fonctionnait dans différents cas de figure, nous avons pu optimiser le temps de parcours en augmentant la vitesse déplacement, la vitesse de rotation, et la vitesse de déploiement du bras. Nous avons également pu gagner du temps en optimisant la stratégie.

i) Tests de fiabilité

Objectifs	• Obtenir un robot fiable.

Mises en situation / Tâches	Résultats
Tester 5 fois chaque cas d'utilisation (pourcentage de réussite).	Non réalisé, pas assez de temps pour faire beaucoup de tests.

Conclusion	• Malgré un bon déroulement des différents tests, nous n'avons pas eu assez de temps pour tester le robot à plusieurs reprises dans les mêmes situations afin de déterminer sa fiabilité. • Nous n'avons également pas testé le robot dans des situations plus complexes (support derrière la zone de récupération, support proche de la zone de récupération).

6 Conclusion

Ce projet pour l'avant projet spatial a été l'occasion de faire une réalisation concrète d'une mission spatiale : l'exploration martienne. Certes, le sujet que nous avons traité est très simplifié (dans les missions du robot et dans le terrain). Cependant, nous avons pu toucher du doigt les problématiques de du robot « autonome » : les contraintes structurelles, les contraintes temporelles, la robustesse, la rapidité, la prise en compte du terrain, la légèreté.

Les délais très courts nous a obligé à obtenir un résultat malgré une méconnaissance initiale des capacités du matériel en notre possession, et du logiciel utilisé. Arrivé à notre but un temps aussi court nous a procuré une réelle satisfaction.

La compétition nous a permis de présenter notre travail. Sur les trois essais, le premier a été un échec dû à un petit problème de robustesse (la pince du robot s'est placée exactement en face du support de l'échantillon, ce qui a bloqué cette dernière). Le deuxième a été une réussite avec un temps de 52 secondes (meilleur temps de tous les robots participants à la compétition). Enfin, la troisième tentative nous a confronté à une mauvaise fiabilité du senseur ultrason (le robot a récupérer une boule « virtuelle ». Vous pouvez trouver les vidéos des trois tentatives de la compétition aux adresses suivantes :

http://www.youtube.com/watch?v=oeumgvPmFxU&feature=youtu.be

http://www.youtube.com/watch?v=y8KFakOKSDw&feature=youtu.be

http://www.youtube.com/watch?v=0JCA3ajPJFM&feature=youtu.be