Charlotte BRICOUT
Charlotte.bricout@polytech-lille.net

Nathan MARTIN
Nathan.martin@polytech-lille.net

Informatique-Microélectronique-Automatique
Ecole Polytechnique Universitaire de Lille

Année universitaire 2014-2015

Rapport de Projet de fin d'études

Modélisation d'un robot chirurgical déformable pour la simulation et le contrôle

---

Tuteur pédagogique : M. Jérémie DEQUIDT

Remerciements

Nous tenons à remercier tout particulièrement M. Christian Duriez, chef de l'équipe Defrost, de nous avoir accueilli au sein d'INRIA.

Nous remercions également M. Jérémie Dequidt, notre tuteur pédagogique, pour son soutien et son encadrement durant le projet.

Nous tenons à remercier M. Mario Sanz-Lopez et Mme Eulalie Coevoet pour la qualité de leurs conseils et l'aide qu'ils nous ont prodigué tout au long de ce projet.

Nous remercions enfin toute l'équipe Defrost pour son accueil et sa disponibilité.

Table des matières

Introduction

Dans le cadre de notre cinquième année à Polytech Lille en spécialité Informatique-Microélectronique-Automatique, nous avons réalisé un projet de fin d'études. Ce projet, nous l'avons mené en collaboration avec l'équipe Defrost de l'Inria Lille ainsi qu'avec l'équipe du centre de recherche en robotique du King's College de Londres. En effet, les deux équipes ont entamé un partenariat sur un projet de création et de contrôle de robot déformable à visée chirurgicale. Le King's College possède actuellement un prototype réel qu'il souhaite commander par le biais d'une simulation informatique. Ce sera alors la simulation qui mettra en mouvement le robot. Les deux équipes nous sollicitent pour implémenter le prototype en simulation sur la plate-forme logicielle SOFA ainsi que pour réaliser son contrôle pneumatique. Ce rapport présente le déroulement du projet et permet de suivre la progression de notre travail. Il expose les résultats obtenus et les perspectives futures du projet. Dans le cadre de notre cinquième année à Polytech Lille en spécialité Informatique-Microélectronique-Automatique, nous avons réalisé un projet de fin d'études. Ce projet, nous l'avons mené en collaboration avec l'équipe Defrost de l'Inria Lille ainsi qu'avec l'équipe du centre de recherche en robotique du King's College de Londres. En effet, les deux équipes ont entamé un partenariat sur un projet de création et de contrôle de robot déformable à visée chirurgicale. Le King's College possède actuellement un prototype réel qu'il souhaite commander par le biais d'une simulation informatique. Ce sera alors la simulation qui mettra en mouvement le robot. Les deux équipes nous sollicitent pour implémenter le prototype en simulation sur la plate-forme logicielle SOFA ainsi que pour réaliser son contrôle pneumatique. Ce rapport présente le déroulement du projet et permet de suivre la progression de notre travail. Il expose les résultats obtenus et les perspectives futures du projet.

1. Contexte et cadre du projet

1.1 L'Institut National de Recherche en Informatique et Automatique

1.1.1 Présentation de l'Inria

L'Institut National de Recherche en Informatique et en Automatique (Inria) est un institut de recherche public français qui oriente ses recherches dans les domaines de l'informatique et les mathématiques des sciences du numérique. Les activités scientifiques d'Inria sont regroupées en cinq domaines de recherche :

- STIC (Sciences et Technologies de l'Information et de la Communication) pour les sciences de la vie et de l'environnement
- Mathématiques appliqués, calcul et simulation
- Perception, cognition et interaction
- Réseaux, systèmes et services, calcul distribué
- Algorithme, programmation, logiciels et architectures

L'institut est composé de 8 centres de recherche répartis dans toute la France et nous travaillerons, durant ce projet, avec le centre Lille - Nord Europe.

1.1.2 Le centre Inria Lille - Nord Europe

Le centre de recherche Inria Lille - Nord Europe, situé à la Haute Borne, compte 17 équipes-projet. Celles-ci se concentrent sur les Sciences et Technologies de l'Information et de la Communication. Elles conçoivent des logiciels innovants pour le commerce ou la logistique, ou encore, mettent au point des simulateurs médicaux ou des interfaces facilitant l'interaction entre l'humain et la machine. Elles orientent leurs recherches sur quatre grandes thématiques :

- Internet des données et Internet des objets
- Génie logiciel pour systèmes éternels
- Modèle patient dynamique
- Couplage perception/action pour l'interaction homme-machine

1.1.3 L'équipe-projet Defrost

Parmi les dix-sept équipes du centre de Lille, l'équipe Defrost oriente ses recherches sur le thème Images, modèles et algorithmes pour la médecine et les neurosciences. Le rôle de cette équipe-projet est de mener des recherches sur des problèmes scientifiques liés à la simulation médicale numérique. Pour ce faire, ils utilisent une plate-forme logicielle : SOFA et collaborent avec des instituts chirurgicaux. L'objectif commun des projets de l'équipe est de fournir des outils de diagnostics, de planification ou de guidage basés sur la simulation physique.

1.2 Le King's College de Londres

1.2.1 Présentation du King's College

Le King's College de Londres est un établissement d'enseignement supérieur britannique et cofondateur de l'Université de Londres. Il est également un membre fondateur du Russell Group, qui constitue le plus grand centre dédié à l'éducation des professionnels de la santé en Europe. Ce complexe regroupe six centres de conseil en recherche médicale. Aujourd'hui, Le King's College est organisé en neuf écoles réparties en quatre campus à Londres.

L'équipe Defrost est en collaboration avec le centre de recherche en robotique (Centre of Robotics Research).

1.2.2 Le centre de recherche en robotique

Les activités du centre de recherche en robotique portent sur :

- la robotique et automatisation
- la chirurgie et la réadaptation robotique
- les systèmes robotisés médicaux
- la manipulation des systèmes robotiques
- la cinématique et les mécanismes
- l'art mimétique inspiré des mécanismes et des robots
- le développement de systèmes d'inspection et de surveillance
- les systèmes biomimétiques, neuraux et cognitifs

Bon nombre des projets de recherche sont inter-disciplinaires et le centre entretient de nombreux partenariats avec des entreprises industrielles et des universités telles que The Imperial College, Newcastle, Birmingham ou encore l'Université de Birkbeck de Londres.
Defrost entretient une collaboration étroite avec l'équipe dirigée par le professeur Kaspar Althoefer. Cette équipe a pour thématique de recherche :

- *Robotics and Intelligent Systems*, Robotique et systèmes intelligents
- *Embedded Sensing and Classification*, Classification de systèmes intégrés
- *Medical and Industrial Applications*, Applications médicales et industrielles

1.3 Présentation du projet

Les équipes Defrost et du centre de recherche en robotique vont travailler pendant deux ans sur ce projet de robot déformable. Actuellement, les robots se constituent de squelettes rigides dont les articulations sont mises en mouvement par des moteurs. Cependant, il est également possible de mettre une structure en mouvement en la déformant. Les robots déformables présentent des avantages non négligeables par rapport aux robots rigides : leur flexibilité, leur robustesse, leur prix de fabrication ainsi que leur durabilité.
L'objectif de ce projet est de contrôler des robots non-rigides qui pourraient faire progresser la chirurgie mini-invasive [1]. A terme, le laboratoire envisage de créer un endoscope pneumatique. Actuellement, le King's College possède un prototype destiné à un contrôle pneumatique. En effet, le contrôle est effectué en emplissant ou vidant des volumes d'air. Le projet pourrait évoluer avec un contrôle hydraulique qui serait plus précis grâce à une mesure du liquide injectée dans les cavités.

On se propose d'intégrer le projet afin de réaliser le contrôle pneumatique du prototype par le biais d'une simulation temps réel grâce à la plate-forme logicielle SOFA.
Pour cela, on se doit de remplir les objectifs suivants :

- Développer un modèle 3D du robot et le mailler en éléments finis grâce à un logiciel de Conception Assistée par Ordinateur
- Mettre en place une simulation temps réel SOFA
- Faire un prototype pneumatique simplifié de l'endoscope afin de valider le contrôle du robot par le biais d'une simulation informatique. Pour mener à bien le projet, une partie de contrôle bas-niveau des régulateurs de pression sera nécessaire
- Réaliser le contrôle pneumatique du prototype
- Mettre à jour le plug-in SOFA pour la communication série entre la simulation informatique et le prototype relié au dispositif de contrôle pneumatique
- Interfacer la simulation avec le prototype réel. C'est-à-dire permettre le contrôle du robot par le bais de la simulation informatique
- Faire une étude de faisabilité pour passer de la technologie pneumatique à hydraulique

1. Procédures chirurgicales conçues pour minimaliser la gêne des patients et le temps de rétablissement.

Notre système est constitué de la façon suivante :

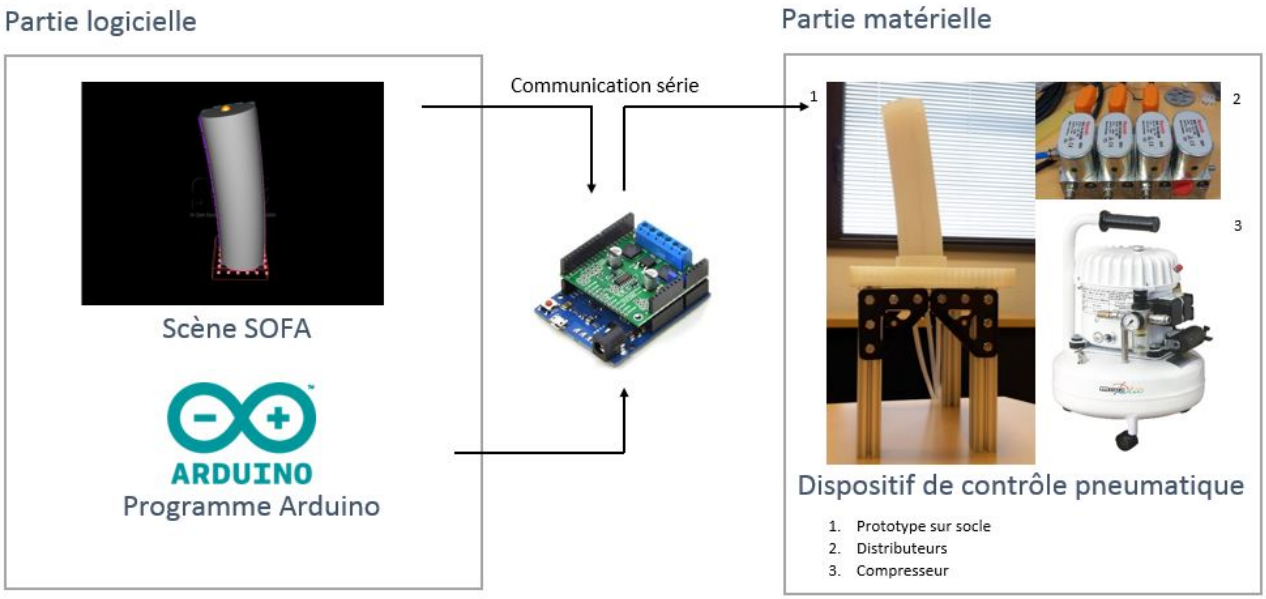

FIGURE 1.1 – Schéma global du système

Grâce à une scène SOFA et à la mise à jour du plug-in SoftRobot, nous pouvons déformer des modèles 3D dans une simulation temps réel. Les données conduisant à cette déformation vont être envoyées à une carte arduino. Ce circuit imprimé va non seulement récupérer les données mais aussi les transmettre au dispositif de contrôle pneumatique. Ainsi, la communication entre SOFA et Arduino comprendra les trois pressions à injecter dans les cavités du prototype, afin de contrôler le robot déformable réel.

2. Contraintes matérielles

Nous souhaitons dans un premier temps réaliser un modèle du prototype. C'est à dire créer par le biais d'un logiciel de CAO un modèle en 3 dimensions et le mailler. Le maillage est une modélisation géométrique du modèle par des éléments finis d'une certaine forme géométrique (Ici, nous utiliserons des tétraèdres). Il est utile pour simplifier un système dans l'optique de simulations, de calculs ou de représentations graphiques.
Pour créer notre modèle, nous avons à disposition les logiciels Rhinoceros 3D et GID.

2.1 Le logiciel GID

GID est un logiciel adaptatif universel pré et post-processeur pour les simulations numériques. Il a été conçu pour couvrir tous les besoins en simulation numérique de pré à post-traitement : modélisation géométrique, définition efficace des données d'analyse, maillage, transfert de données à un logiciel d'analyse et visualisation des résultats numériques.
Afin de mettre en simulation le modèle 3D, nous utilisons SOFA, décrite ci-dessous.

2.2 Le logiciel Rhinoceros 3D

Rhinoceros 3D est un logiciel de Conception Assistée par Ordinateur utilisé essentiellement pour concevoir des formes complexes. Il permet de travailler de façon libre sur des courbes et des surfaces. Rhino est un modeleur surfacique qui utilise des polysurfaces fermées. C'est aussi un mailleur puissant pour la conversion de fichiers ou pour la visualisation.

Nous allons ensuite utiliser la plate-forme logicielle SOFA. Nous insérons notre prototype 3D dans une scène compatible à SOFA qui permettra de jouer une simulation informatique.

2.3 La plate-forme logicielle SOFA

SOFA est une plate-forme de recherche et développement dédiée aux simulations physiques interactives et en particulier à la simulation médicale. La plate-forme regroupe de nombreux algorithmes issus de domaines de recherche variés. Des modèles de nature différente peuvent être combinés de façon à produire des simulations complexes qui restent compatibles avec des temps de calcul optimaux.

SOFA
Simulation Open Framework Architecture

D'un point de vue hardware, nous allons utiliser une carte Arduino qui permettra l'envoi de données entre la simulation informatique et le prototype réel raccordé à un dispositif de contrôle pneumatique.

2.4 Le logiciel Arduino

Arduino est un circuit imprimé en matériel libre sur lequel se trouve un microcontrôleur qui peut être programmé pour analyser et produire des signaux électriques, de manière à effectuer des tâches très diverses comme le pilotage d'un robot. C'est une plate-forme basée sur une interface entrée/sortie simple. Le logiciel de programmation des modules Arduino est une application Java multi-plateforme servant d'éditeur de code et de compilateur. Celui-ci permet de transférer le programme au travers de la liaison série. Le langage de programmation utilisé est le C++, lié à la bibliothèque de développement Arduino, permettant l'utilisation de la carte et de ses entrées/sorties.

3. Réalisations techniques

3.1 Contrôle des robots

3.1.1 La commande des robots rigides

Un système articulé rigide est caractérisé par une structure arborescente articulée simple ou multiple dont les liaisons sont mobiles les unes par rapport aux autres. Ce système a pour objectif de mener l'organe terminal vers un lieu géométrique imposé par la tâche.

La synthèse de la commande du robot nécessite la connaissance des relations entre ses grandeurs d'entrées et de sorties. L'ensemble de ces équations constitue le modèle mathématique du robot. Si les équations sont extraites de la physique, le modèle est appelé modèle de connaissance, et si ces équations découlent des observations disponibles sur le système, le modèle s'appelle modèle de représentation.

Les modèles dynamiques des bras manipulateurs sont décrits par un ensemble d'équations mathématiques qui portent des informations dynamiques de ces manipulateurs. Ils peuvent être simulés sur un ordinateur dans le but de synthétiser une commande conditionnée par des performances désirées. L'ensemble des équations dynamiques peut être déterminé par des lois mécaniques classiques Newtoniennes et Lagrangiennes. Les approches d'Euler Lagrange et Newton-Euler permettent d'aboutir aux équations du mouvement des robots.

3.1.2 La commande de robot déformable

Les robots déformables possèdent un grand nombre de degrés de liberté. Il faut, de ce fait, aborder leur contrôle d'une toute autre manière. En effet, il faut intégrer une modélisation précise des déformations dans le processus de contrôle. Pour cela, il est nécessaire d'utiliser des méthodes numériques précises telles que les éléments finis (FEM). Étant donné le grand nombre de degrés de liberté des modèles, il est impossible d'utiliser les méthodes classiques de contrôle robotique.
Nous allons donc créer un modèle 3D par CAO, qui sera maillé en éléments finis. Nous intégrerons ce modèle sur la plate-forme logicielle SOFA où nous utiliserons la FEM temps réel.

FIGURE 3.1 – Les différents types de commande

3.2 Elaboration d'un modèle pour la simulation

3.2.1 Modélisation du prototype

Grâce à une publication, on connaît les dimensions et spécifications du robot. Celui-ci est représenté sur la figure ci-dessous.

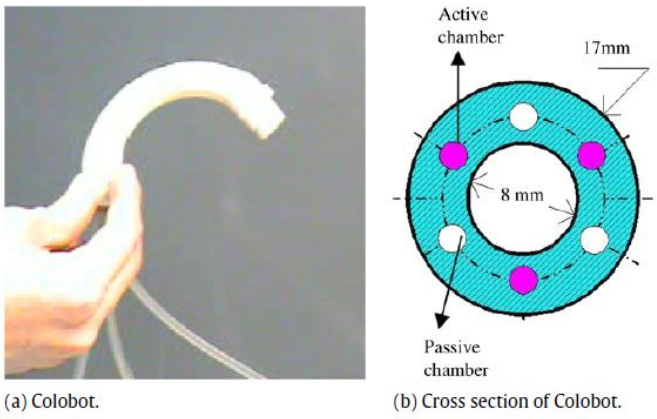

FIGURE 3.2 – Robot déformable et sa section

Le robot est constitué de 6 cavités très fines mais seules trois cavités recevront un volume d'air. Dans un premier temps, nous avons utilisé le logiciel Blender pour concevoir le prototype. Nous avons générer un modèle au format ".obj" afin d'effectuer le maillage sur SOFA. Mailler une forme conçue par Conception Assistée par Ordinateur (CAO) signifie la représenter géométriquement par des éléments finis. Ici, nous avons utilisé des tétraèdres.
Nous nous sommes aperçus que le prototype modélisé sous Blender contient des cavités trop fines pour être maillées sur le logiciel SOFA. Le nombre de noeuds du prototype est trop important et nos machines ne supportent pas les calculs. De plus, un nombre de tétraèdres moins important ne fait pas apparaître les cavités sur le prototype maillé. Après plusieurs tentatives de maillage vaines sur Blender, nous avons décidé de travailler avec le logiciel GID. Celui-ci permet la représentation du prototype ainsi que son maillage.

La forme particulière du prototype nous oblige à porter une attention toute particulière au maillage des cavités. Nous avons donc élaboré deux modèles :

— un modèle comportant 2000 à 3000 noeuds qui permettra une simulation temps réel
— un modèle comportant 10 000 à 11 000 noeuds qui permettra d'évaluer les besoins en pression.

Cependant, même en prenant soin de mailler plus précisément les cavités par rapport au reste du modèle, le maillage n'est pas non plus concluant avec GID. Finalement, nous décidons, en accord avec le King's College, de créer un prototype deux fois plus large que celui prévu initialement. Ainsi, les cavités sont plus simples à mailler et nous générons les deux prototypes souhaités.

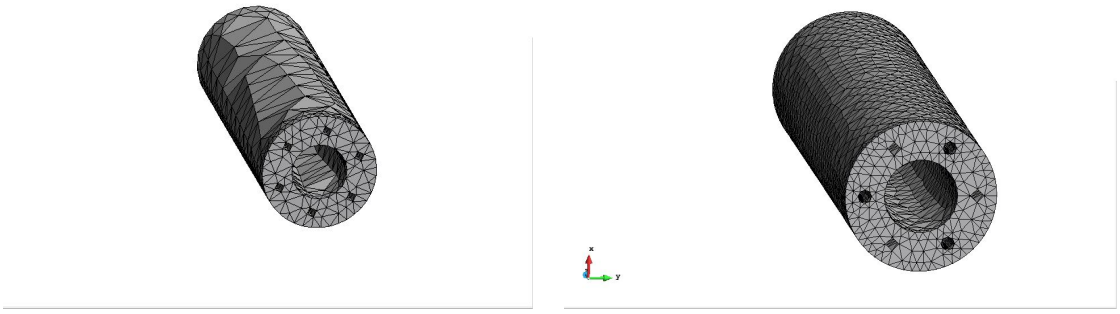

FIGURE 3.3 – Modèle à 2600 noeuds et modèle à 10 000 noeuds

3.2.2 Mise en simulation avec SOFA

On souhaite, à présent, mettre en simulation notre modèle exporté de GID au format ".msh". Ce fichier mesh contient les coordonnées des points ainsi que les éléments de l'objet maillé. A partir de ce mesh, on cherche à générer un fichier au format ".xml" afin que les données contenues dans le mesh soient exploitables dans une scène Sofa. Pour ce faire, on utilise le script suivant qui génère le fichier souhaité :

```
awk 'BEGIN { print "<Mesh position=\""; section=0 } /Coordinates/ { section=1 } (
NF==4&&section==1) { print $2 " " $3 " " $4 } /Elements/ { section+=1; if (
section==2) { print "\"\n tetrahedra=\"" } } (NF==5&&section==2) { print $2-1 " "
 $3-1 " " $4-1 " " $5-1 } END { print "\" />" }'
```

On souhaite, dans un premier temps, valider notre modèle et le contrôler grâce à la technologie pneumatique. On crée donc une scène SOFA (Voir Annexe 1) qui permettra de jouer une simulation. SOFA permet le contrôle d'un modèle par le biais d'une interface graphique. Sur cette interface, on peut manuellement définir une trajectoire pour le robot en utilisant la souris. Grâce à cette scène, on calcule la quantité d'air à injecter dans les cavités pour obtenir le déplacement demandé.
SOFA utilise la méthode des éléments finis (FEM) qui permet de résoudre de manière discrète des équations différentielles partielles dont on cherche une solution approchée. Cette méthode directe s'appuie sur le principe fondamental de la dynamique :

$$\frac{M\partial^2 x}{\partial t^2} + F(\frac{\partial x}{\partial t}, x) = Fext$$

Dans cette méthode, on connaît les forces appliquées au modèle et on cherche la position qu'il atteindra sous la contrainte de ces forces. Notre problématique est légèrement plus complexe car ici, on cherche à connaître la force à appliquer pour atteindre une position donnée. On utilise pour cela une méthode inverse temps réel qui repose sur les éléments finis non-linéaires, permettant une estimation du module de Young. En effet, on ajoute à l'équation précédente des contraintes :

$$\frac{M\partial^2 x}{\partial t^2} + F(\frac{\partial x}{\partial t}, x) = Fext + \lambda H^T$$

On utilise alors un algorithme itératif d'analyse numérique afin d'estimer la valeur de la contrainte. Cette méthode repose sur une projection du modèle dans l'espace des variables d'optimisation. On valide la méthode par des exemples numériques que l'on applique au modèle. [1]

1. Explications tirées de Introducing interactive inverse FEM simulation and its application for adaptive radiotherapy Coevoet E., Reynaert N., Lartigau E., Schiappacasse L., Dequidt J. & Duriez C. in Medical Image Computing and Computer Assisted Intervention, pages 81-88 2014

Dans notre scène, on cherche tout d'abord à sélectionner les trois cavités qui nous intéressent et pour ce faire, on utilise des 'BoxROI'. Les 'BoxROI' sont des boîtes permettant de sélectionner les tétraèdres d'un volume. Ici, on en créé quatre : la première permet de fixer la base du robot et les trois autres permettent de sélectionner les 3 cavités à contrôler. Une fois que les tétraèdres sont sélectionnés, on récupère tous les triangles qui composent les cavités. Cette opération permet de sélectionner la surface au lieu du volume et nous permettra le contrôle par pression. On fixe alors un point que l'on nomme "Goal" et un second point nommé "Effector". "Goal" est le point que va suivre le modèle."Effector" est un point du modèle qui cherchera à converger vers le "Goal". Une fois ces deux points définis, il nous reste à tester différentes pressions pour connaître la tolérance de notre système ainsi que plusieurs valeurs du module de Young pour avoir la rigidité souhaitée.

Pour mettre en simulation le modèle à 10 000 noeuds, nous ne pouvons pas directement jouer la scène SOFA. En effet, le modèle étant très précis, nous ne pouvons pas utiliser une simulation temps réel car notre machine ne calcule pas assez vite. Nous avons donc programmé une trajectoire pour la sphère "Goal" et avons réalisé des captures de la simulation à chaque pas de calcul. Il nous suffit ensuite de convertir l'ensemble des photos en une vidéo. Nous avons, pour cela, modifié le fichier scène afin qu'il génère un fichier texte contenant les coordonnées de la sphère "Goal" durant la simulation. Il nous a ensuite fallu modifier le fichier texte pour y coder la trajectoire désirée et le fichier scène pour qu'il la lise. Ainsi lors du déroulement de la simulation, la sphère adoptera successivement les positions définies dans ce fichier texte et notre robot essaiera de rapprocher son centre outil de celle-ci.
Pour transformer l'ensemble des images obtenues en une vidéo, nous avons utilisé le logiciel Avidemux. Ainsi, on détermine une trajectoire circulaire, par exemple, et on observe le résultat en vidéo.

Pour le modèle à 2600 noeuds, on constate que la simulation n'est pas assez précise pour être utilisée ici. De ce fait, il nous faut trouver une alternative à ce modèle qui pourra être jouée en temps réel. En ce qui concerne le modèle à 10 000 noeuds, le résultat obtenu en vidéo n'est pas satisfaisant. En effet, on constate que la pression dans les cavités n'est pas uniforme (voir l'image ci-dessous) et ceci est dû aux maillages des cavités.

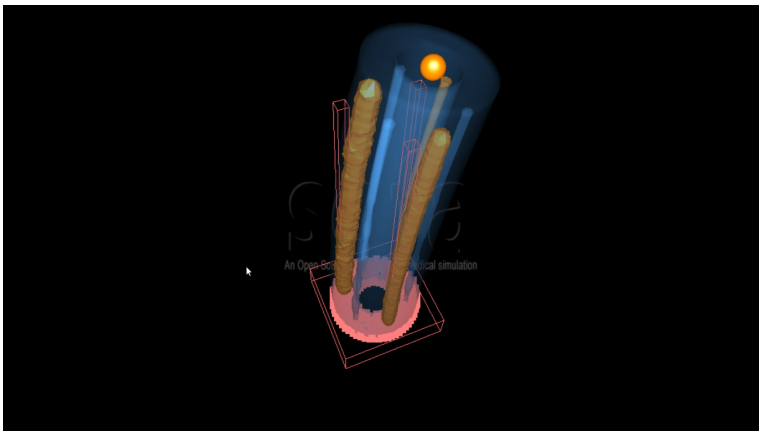

FIGURE 3.4 – Robot simulé sous pression

Finalement, on décide d'utiliser le logiciel Rhinoceros 3D afin d'extraire directement un fichier "objet" maillé. Ce logiciel permet d'obtenir un fichier compatible à notre scène SOFA avec un maillage automatique moins lourd que celui que propose SOFA.

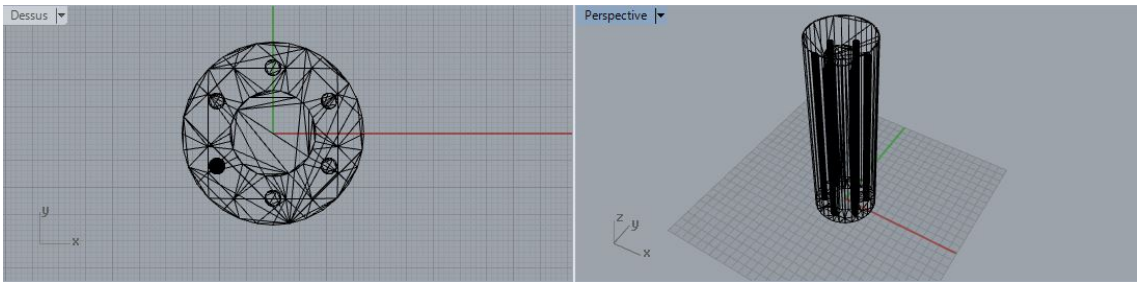

FIGURE 3.5 – Modélisation 3D sur Rhinoceros

Dans la scène, on procède différemment pour contrôler les cavités. On exporte séparément chacune des cavités avec Rhinoceros et on exporte également le prototype dans sa totalité. Ainsi, les cavités sont directement identifiées et nous n'utilisons donc plus les "BoxROI". Pour finir, on obtient une simulation temps réel grâce à cette dernière alternative. Cependant, on observe sur la première figure ci-dessous que le robot ne se déforme pas de façon réelle. En effet, il reste droit et n'adopte pas de forme courbée comme dans la réalité. Nous nous sommes donc aperçus que le modèle ne comportait pas de section intermédiaire au sein de sa structure. Il faut alors le modifier à partir du fichier ".obj" afin de lui ajouter des sections et donc, des points supplémentaires à prendre en compte en simulation. Nous obtenons donc la seconde figure ci-dessous :

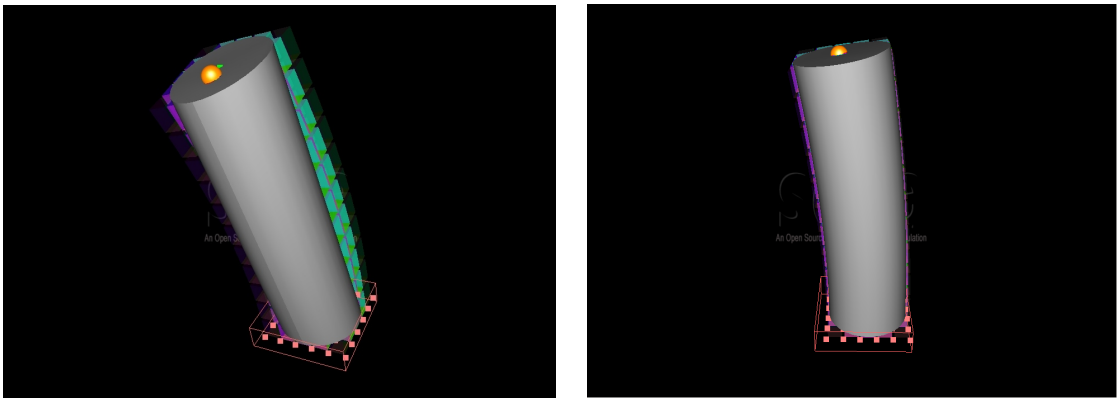

FIGURE 3.6 – Simulation temps réel du prototype

3.3 Création d'un prototype

Nous sommes chargés de réaliser un premier prototype du robot déformable car nous n'avons pas à disposition le prototype du King's College. Pour ce faire, il nous faut concevoir un moule pour couler du silicone et obtenir la géométrie souhaitée. Nous réaliserons ce moule à partir de tuyaux pour la forme du robot et de tiges filetées pour délimiter les cavités.

FIGURE 3.7 – Moule pour le prototype

Dans ce moule, nous allons couler du silicone et laisser prendre pour enfin démouler la première partie du prototype. La seconde partie est un disque de 1 cm d'épaisseur qui vient fermer le haut des cavités pour empêcher l'air de passer.
Nous avons réalisé ce processus de fabrication à plusieurs reprises en utilisant plusieurs types de silicone. En effet, nous avons réalisé plusieurs prototypes en silicone Ecoflex 00-30 et un prototype en silicone Dragon Skin A10. L'Ecoflex 00-30 est un silicone très mou alors que le Dragon Skin A10 est plus rigide. Ainsi, nous avons pu réaliser des tests sur des prototypes qui se déforment plus ou moins fortement. Nous avons réalisé deux prototypes identiques afin de vérifier que la déformation est identique. De cette façon, on valide le processus de fabrication.

Une fois le premier prototype opérationnel, on conçoit un socle sur lequel le fixer. On utilise pour cela un OpenBeam Precut Kit. Il est également nécessaire de créer une pièce sur mesure pour maintenir le prototype à sa base. On réalise donc une pièce par Conception Assistée par Ordinateur et on l'imprime par le biais d'une imprimante 3D. Enfin, on perce une plaque pour y faire passer les 3 tuyaux nécessaires au contrôle pneumatique et on y fixe la pièce conçue pour le maintien du robot.

Finalement, on se doit d'apporter quelques modifications au prototype en ce qui concerne le maintien des tuyaux. En effet, pour maintenir les tuyaux, il est nécessaire de les rentrer en force dans le silicone. Pour cela, on rajoute un disque de silicone à l'extrémité encore ouverte du prototype. Enfin, on perce finement ce disque et on peut insérer les tuyaux.

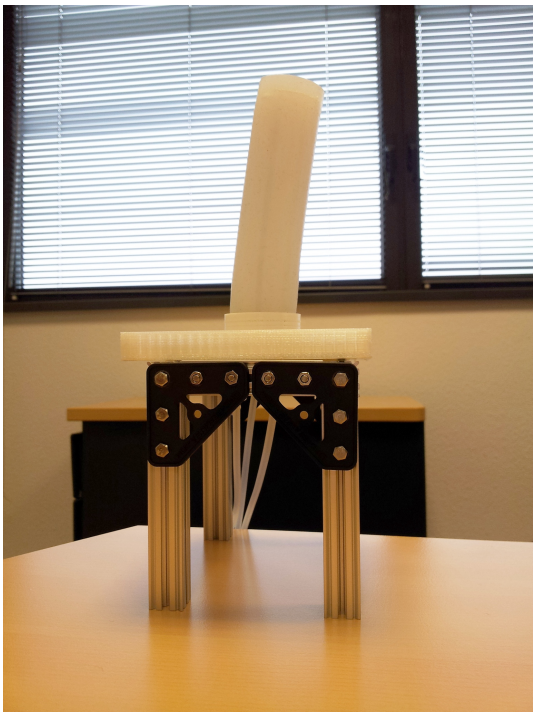

FIGURE 3.8 – Socle du prototype

3.4 Tests réels en pression

Nous avions accès à un dispositif de contrôle pneumatique composé d'un compresseur pneumatique, de trois distributeurs et de tuyaux à relier aux cavités. Le compresseur sert à fournir de l'air à haute pression. Quant aux distributeurs, Ils ont pour fonction de distribuer l'air dans les cavités.
Dans un premier temps, il nous a fallu réaliser le contrôle du prototype indépendamment de la simulation informatique. La commande est transmise par le biais d'une carte arduino et d'une carte électronique, reliées au dispositif décrit précédemment. Nous avons donc créé un programme arduino permettant le contrôle et la régulation de chacune des trois cavités. Ce programme permet d'évaluer la valeur maximale de pression que supporte le robot et de tester le bon fonctionnement du dispositif de contrôle.
Grâce à ce programme, nous avons pu tester les différents prototypes et déterminer lequel est le plus adapté à notre application. Le robot retenu est celui composé de silicone Dragon Skin A10. En effet, l'autre prototype était trop flexible et rebondissait lors de ces mouvements, rendant la commande imprécise. Par ailleurs, le prototype retenu est beaucoup moins flexible et nécessite l'envoi de pressions plus importantes.

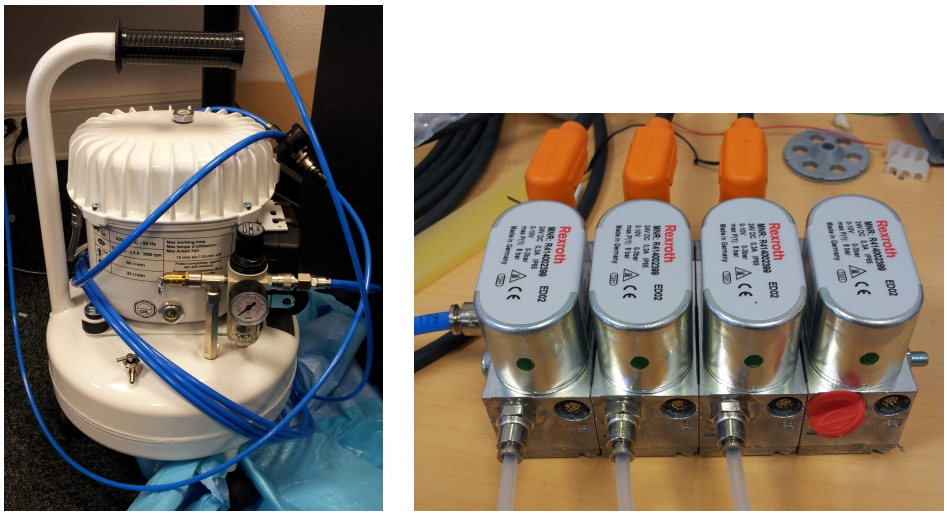

FIGURE 3.9 – Compresseur et distributeurs pneumatiques

3.5 Interfaçage entre la simulation et le prototype

Il est nécessaire d'interfacer le prototype physique avec la simulation pour réaliser le contrôle pneumatique par simulation. C'est à dire, de connecter la partie software avec la partie hardware. L'interfaçage se compose de plusieurs étapes :

- Mettre à jour le plug-in Softrobot pour le contrôle pneumatique
- Adapter la scène SOFA pour la communication série
- Créer un programme arduino pour la communication série

3.5.1 Mise à jour du plug-in SoftRobot

Le plug-in SOFA SoftRobot permet, de manière générale, la simulation de robot déformable. Initialement, ce plug-in permettait de simuler des robots contrôlés par le biais de servomoteurs uniquement. Il a été actualisé pour pouvoir réaliser l'équivalent d'un contrôle pneumatique en simulation. Par ailleurs, les parties communication série et envoi de données à une carte arduino n'ont pas été programmées pour le contrôle pneumatique. De ce fait, nous avons mis à jours ces deux parties pour permettre la communication entre SOFA et la carte arduino. Ainsi, on récupère de SOFA les pressions à appliquer et on les transmet par le biais du porte série. On créé un tableau "paquet" constitué des cinq cases suivantes : [245][0][Pression1][Pression2][Pression3]. Ainsi, on connaît l'emplacement des données à lire et à appliquer à notre prototype.

3.5.2 Adaptation de la scène SOFA

La scène SOFA doit être actualisée pour prendre en compte la communication série. On indique les informations nécessaires à la communication telles que le nom du port, l'emplacement des données à envoyer et le nombre de bits à transmettre par seconde (Baud Rates).

3.5.3 Création d'un programme arduino

Enfin, il nous faut créer un programme arduino permettant de lire les données de la simulation et de les envoyer au prototype. Une fois la suite de chiffres [245][0] détectée, on lit et on envoie directement les pressions dans les cavités.

4. Résultats finaux et perspectives d'amélioration

4.1 Résultats finaux

Après avoir franchi avec succès les différentes étapes décrites précédemment, nous devons vérifier que les données envoyées sont les mêmes que données reçues. Nous avons donc modifié le code afin d'obtenir un affichage du transfert des informations. Les informations qui transitent entre SOFA et Arduino sont effectivement identiques. Afin de mettre au point une démonstration pour le King's College, nous avons programmé une trajectoire automatique de la sphère "goal" . Pour mettre en évidence des mouvements selon plusieurs axes, nous avons choisi une trajectoire circulaire. Pour cela, nous avons rédigé un script shell écrivant dans un fichier ".ws", les coordonnées de la sphère pour chaque pas de temps. Ainsi, nous pouvons choisir de diriger le modèle avec la souris dans SOFA ou indiquer une trajectoire à exécuter automatiquement. Cette trajectoire est insérée dans la scène SOFA et elle est lue au démarrage de la simulation informatique.
D'un point de vue matériel, le prototype et son dispositif de contrôle sont opérationnels. Par ailleurs, des changements sont à prévoir pour la migration technologique de pneumatique à hydraulique.

4.2 Etude de faisabilité de migration de technologie de contrôle

Un contrôle hydraulique du robot déformable a été envisagé dans le but d'obtenir plus de précision dans ses mouvements. L'eau est un liquide non compressible que l'on va injecter dans les cavités pour obtenir la déformation voulue. On connaîtra alors précisément la quantité d'eau insérée dans l'une ou l'autre des cavités et on pourra être plus précis quand au mouvement du robot déformable.

Du point de vue de la simulation, peu de changements sont à prévoir. Dans SOFA, on cherchait initialement à estimer la pression conduisant à la déformation du modèle en utilisant la méthode des éléments finis. A chaque pas de temps de la simulation, les forces internes au modèle sont linéarisées :

$$f(xi) \approx f(xi-1) + K(xi-1)dx$$

Avec f : les forces internes en un point x donné
K(x) : la matrice de raideur
dx : la différence de positions entre deux pas de temps.

A chaque étape, on résout l'équation d'équilibre statique des forces :

$$-K(xi-1)dx = p + f(xi-1) + L^T\lambda$$

Avec p : les forces externes connues (par exemple, la gravité) $J^T\lambda$: la contribution des multiplicateurs de Lagrange qui sont définis en deux types :

- Les actionneurs λa qui sont les efforts externes à déterminer
- Les effecteurs λe qui décrivent la déformation

Ici, l'actionneur à déterminer est la pression. La prochaine étape consiste à projeter les équations du modèle FEM dans l'espace des contraintes. On procède alors de la façon suivante :

1. En résolvant l'équation de l'équilibre statique en posant $\lambda = 0$, on obtient une configuration libre x^{free} du modèle déformable, ainsi que δe^{free} la violation de la contrainte sur l'effecteur. Par exemple, pour un contrôle en position, on récupère le vecteur entre la position de l'effecteur pendant la configuration libre et la position à atteindre.
2. On projette le système dans l'espace des contraintes, ce qui permet d'obtenir le problème inverse le plus petit possible :
$$\delta e = JeK^{-}1Ja^T\lambda a + \delta^{free}$$
Avec δe : le vecteur entre les positions réelles de l'effecteur et la position à atteindre.

3. On corrige la configuration finale, à la fin du pas de temps.

$$x = x^{free} + K^{-1}Ja^T\lambda a$$

Pour passer en hydraulique, on conserve un actionnement en pression dans la simulation. Cependant on adaptera la donnée de sortie à transmettre à l'actionneur hydraulique. En sortie, sera donc envoyé un volume et non plus une pression. Il suffira de calculer les variations de volume de la cavité (calcul fait à partir du maillage surfacique de la cavité) à chaque pas de temps de la simulation.

4.3 Problèmes rencontrés

L'aspect le plus contraignant de ce projet a été l'accès aux bureaux de l'équipe. Notre statut n'étant pas officiel pour des raisons administratives, il nous était, parfois, difficile de travailler à l'Inria. Nous avons cependant bénéficié d'une équipe conciliante et le projet a pu se dérouler correctement, nous permettant d'atteindre les objectifs initiaux.
La seconde difficulté de ce projet de fin d'études réside dans sa répartition temporelle. En effet, étant réparti sur plusieurs mois, nous avons dû faire face à un nombre d'heures allouées au projet très variable d'une semaine à l'autre. De ce fait, nous avons dû gérer plusieurs interruptions dans notre travail. Cet inconvénient nous a par ailleurs obligé à une organisation solide et à un travail rigoureux.

Conclusion

Tous les objectifs du projet ont été atteints et nous avons pu présenter nos résultats au centre de recherche du King's College qui en est satisfait. Actuellement, nous sommes capables de commander un robot déformable, mis en mouvement par un dispositif pneumatique, par le biais d'une simulation informatique. On peut le commander directement par le biais de l'interface graphique de la plate-forme logicielle SOFA en utilisant le pointeur de la souris. On peut également définir une trajectoire qui sera lue au démarrage de la simulation.
Le projet va pouvoir évoluer avec un contrôle hydraulique qui se substituera à la commande pneumatique. Il sera nécessaire de revoir les proportions du prototype afin que celui-ci soit conforme aux dimensions d'un endoscope. Il faudra également changer sa composition pour permettre son utilisation en chirurgie mini-invasive.
A travers ce projet, nous avons progressé en programmation C++ sur les logiciels Arduino et SOFA. De plus, nous avons appréhendé les grands principes de fonctionnement de la simulation numérique appliquée aux robots déformables. Nous avons également découvert le fonctionnement d'un dispositif de contrôle pneumatique et l'avons pris en main. Nous avons expérimenté plusieurs processus de fabrication de prototype en silicone. Enfin, nous avons progressé en Conception Assistée par Ordinateur en modélisant le prototype sur différents logiciels.
Ce projet est une belle opportunité pour travailler sur un projet conséquent, réparti dans le temps. Cela nous permet, par ailleurs, de sortir du cadre de Polytech Lille pour travailler dans une autre structure et d'être intégrés à une équipe.

5. Annexes

5.1 Annexe 1 : Scène SOFA

```
<?xml version="1.0" ?>
<Node name="root" dt="0.01" gravity="0 -9.81 0">

        <RequiredPlugin pluginName="SoftRobots" />
        <VisualStyle displayFlags="showVisualModels showCollisionModels hideForceFields
showInteractionForceFields showBehaviorModels" />
        <FreeMotionMasterSolver />
        <GenericConstraintSolver maxIterations="500" printLog="1" tolerance="0.000000001"/>
        <CollisionPipeline verbose="0" />
        <BruteForceDetection name="N2" />
        <CollisionResponse response="FrictionContact" />
        <LocalMinDistance name="Proximity" alarmDistance="0" contactDistance="0" />
        <CollisionResponse name="Response" response="FrictionContact" />

        <!-- Interactive Mode -->
        <Node name="goal" activated="1" >
                <EulerImplicitSolver firstOrder="1" />
                <CGLinearSolver iterations="100" />
                <MechanicalObject name="goalMO" position="0 0 110"/>
                <!-- to write or read in a text file in order to move our sphere -->
                <ReadState filename="trajectoire.ws" />
                <Sphere radius="3" group="3"/>
                <UncoupledConstraintCorrection />
        </Node>

        <Node name="VisuGoal" activated="0">
                <OglShader
                    vertFilename="shaders/shaderLibrary.glsl"
                    fragFilename="shaders/shaderLibrary.glsl" />
                <OglFloat3Variable name="AmbientColor" value="0.0 0.0 0.0" />
                <OglFloat3Variable name="DiffuseColor" value="0.8 0.2 0.2" />
                <OglFloatVariable  name="SpecularRoughness" value="0.25" />
                <OglFloatVariable  name="SpecularReflectance" value="0.05" />
                <OglFloat3Variable name="LightColor" value="1 1 1" />
                <OglFloat3Variable name="LightPosition" value="10 10 -60" />
                <OglFloat3Variable name="LightDirection" value="-0.1 -0.1 0.6" />
        </Node>

    <Node name="Cylinder" >
        <EulerImplicitSolver name="cg_odesolver" rayleighMass="1" rayleighStiffness="0.03" />
        <SparseLDLSolver name="preconditioner"   />
        <SparseGridRamification n="6 6 12" fileTopology="mesh/Cylinder.obj" nbVirtualFinerLevels="3"
        finestConnectivity="0" />

        <MechanicalObject name="dofs"  scale="1" dy="2"/>
        <UniformMass totalMass="1.0" />
        <HexahedronFEMForceField name="FEM" youngModulus="15000.0" poissonRatio="0.4"
                                 method="large" updateStiffnessMatrix="false" printLog="0"/>

        <GenericConstraintCorrection solverName="preconditioner" />
        <!-- Modification of box sizes: xmin ymin zmin xmax ymax zmax  box="-0.1 -0.1 -1.001
 0.01 0.01 -0.99" -->

        <BoxROI name="boxROI" box="-20 -20 -5  20 20 5" drawBoxes="true"  />
        <FixedConstraint indices="@boxROI.indices" />

        <Node name="re-fix" activated="true">
                <BoxROI name="boxROI" box="-20 -20 -5  20 20 5" drawBoxes="true"  />
                <FixedConstraint indices="@boxROI.indices" />
        </Node>

        <!-- Cylinder visualization -->
        <Node name="VisuCybot" activated="true">
                        <OglModel name="VisualBody" fileMesh="Cylinder.obj"    />
                        <BarycentricMapping input="@.." output="@VisuCybot" />
        </Node>

        <Node name="Effector" activated="true" >
                <MechanicalObject name="effectorPoint" position=" 0 0 110 " />
                <PositionEffectorConstraintGS index="0" actuator="false"
                effectorGoal="@../../goal/goalMO.position" />
```

```
            <BarycentricMapping name="mapping1"    mapForces="false" mapMasses="false"/>
        </Node>

        <Node name="Cavity1" activated="true">
            <MeshObjLoader filename="Cavity1.obj" name="loader" />
            <Mesh src="@loader" name="topo"/>
            <MechanicalObject name="cavity" />
            <SurfacePressureActuatorConstraintGS name="pressure" triangles="@topo.triangles"
            actuator="true" maxPressure="240" visualization="1" showVisuScale="0.0001"/>
            <Triangle group="3"/>
            <BarycentricMapping name="mapping2"  mapForces="false" mapMasses="false"/>
        </Node>

        <Node name="Cavity2" activated="true">
            <MeshObjLoader filename="Cavity2.obj" name="loader" />
            <Mesh src="@loader" name="topo"/>
            <MechanicalObject name="cavity" />
            <SurfacePressureActuatorConstraintGS triangles="@topo.triangles" actuator="true"
            maxPressure="240" visualization="1" showVisuScale="0.0001"/>
            <Triangle group="3"/>
            <BarycentricMapping name="mapping2"  mapForces="false" mapMasses="false"/>
        </Node>

        <Node name="Cavity3" activated="true">
            <MeshObjLoader filename="Cavity3.obj" name="loader" />
            <Mesh src="@loader" name="topo"/>
            <MechanicalObject name="cavity" />
            <SurfacePressureActuatorConstraintGS triangles="@topo.triangles" actuator="true"
            maxPressure="240" visualization="1" showVisuScale="0.0001"/>
            <Triangle group="3"/>
            <BarycentricMapping name="mapping2"  mapForces="false" mapMasses="false"/>
        </Node>

        <Node name="SerialPortPressureControl" >
            <SerialPortPressureControl port="/dev/ttyACM0" baudRate="115200"
            sentData="@../Cavity1/pressure.sendValues" listening="true"/>
        </Node>

    </Node>
</Node>
```

5.2 Annexe 2 : Programme Arduino

```
/* Charlotte BRICOUT et Nathan MARTIN - IMA 5 Polytech Lille */
/* 05.02.2015 */
/* Control of 3 cavities in order to move a silicone robot */

/*  PIN assignments Serie 1: */

const byte PINsense1 = 0;          // Actuator 1 pressure sense
const byte PINsense2 = 1;          // Actuator 2 pressure sense
const byte PINsense3 = 2;          // Actuator 3 pressure sense
const byte PINpressure1 = 10;      // Actuator 1 pressure command
const byte PINpressure2 = 9;       // Actuator 2 pressure command
const byte PINpressure3 = 8;       // Actuator 3 pressure command

// initialization

int nbCavity=3;
boolean setNumberCavity=true;

int Cavity_Pressure[3];

int header = 0; //to spot the beginning of informations on the serial port
int pressure= 0; //Pressure in cavity at the initialization

int test=0;

#ifndef cbi
#define cbi(sfr, bit) (_SFR_BYTE(sfr) &= ~_BV(bit))
#endif
#ifndef sbi
#define sbi(sfr, bit) (_SFR_BYTE(sfr) |= _BV(bit))
#endif

void setup()
{
  // set ADC prescaler to 16 ->77kHz read
  sbi(ADCSRA,ADPS2) ;
  cbi(ADCSRA,ADPS1) ;
  cbi(ADCSRA,ADPS0) ;

  analogReference(INTERNAL2V56);
  analogRead(PINsense1);
  analogRead(PINsense2);
  analogRead(PINsense3);

  TCCR1B = TCCR1B & 0b11111000 | 0x01;  //Prescaler 1 modified, fPWM=32kHz
  TCCR2B = TCCR2B & 0b11111000 | 0x01;  //Prescaler 2 modified, fPWM=32kHz
  TCCR3B = TCCR3B & 0b11111000 | 0x01;  //Prescaler 3 modified, fPWM=32kHz
  TCCR4B = TCCR4B & 0b11111000 | 0x01;  //Prescaler 4 modified, fPWM=32kHz

  analogWrite(PINpressure1, 0);
  analogWrite(PINpressure2, 0);
  analogWrite(PINpressure3, 0);

  Serial.begin(115200);
  delay(2000);
  /*
  // we send the number of cavity
  int c;
  while(c!=97)
  {
  while(!Serial.available());
  c = Serial.read();
  delay(1);
  if(c!=97)
  {
    Serial.print(-1);
  }
  }

  Serial.print(nbCavity);
  Serial.flush();
  delay(1);
*/
}

void loop()
{
```

```
  if(Serial.available())
  {
   delay(1);
   test=Serial.read();
   while(test!=245)
   {
     delay(1);
     test=Serial.read();
   }

   test=Serial.read();

   while(!Serial.available());
   Cavity_Pressure[0]=0.33*Serial.read();
   delay(1);
   while(!Serial.available());
   Cavity_Pressure[1]=0.33*Serial.read();
   delay(1);
   while(!Serial.available());
   Cavity_Pressure[2]=0.33*Serial.read();
   delay(1);
  }

  Cavity_Pressure[0]=constrain(Cavity_Pressure[0], 1,80);
  Cavity_Pressure[1]=constrain(Cavity_Pressure[1], 1,80);
  Cavity_Pressure[2]=constrain(Cavity_Pressure[2], 1,80);

  analogWrite(PINpressure1,Cavity_Pressure[2]);
  analogWrite(PINpressure2,Cavity_Pressure[1]);
  analogWrite(PINpressure3,Cavity_Pressure[0]);
}
```