Institut des Sciences et Techniques de Grenoble

INRIA RHÔNE-ALPES

BOZONNET Vincent
Elève ingéneur 3ième année
ISTG section 3i option automatique

Programmation d'un robot bipede

Stage du 2 avril au 28 septembre 2001

Maitre de stage : Herve Mathieu

SOMMAIRE

Remerciements

Je tiens à remercier, vivement l'INRIA Rhône-Alpes de m'avoir accueilli pour effectuer mon stage de fin d'études dans le service robotique, et m'offrant ainsi la possibilité d'acquérir une expérience enrichissante.

J'adresse mes remerciements à M. Hervé Mathieu et M. Jean-François Cuniberto, qui m'ont accueilli au sein de son service, et qui m'ont suivi tout au long du stage. Ces personnes ont facilité mon intégration dans les différentes équipes.

Je remercie également toutes les personnes qui m'ont aidé à mettre à bien mon projet.

Vincent BOZONNET
Etudiant en 3ième année à l'ISTG

Introduction

Dans le cadre de la formation en école d'ingénieur à l'ISTG, un travail de fin d'études est effectué par chaque élève de troisième année. Sa finalité est de se familiariser avec le milieu industriel, en abordant des problèmes nouveaux ou en approfondissant des questions importantes pour l'entreprise accueillante (stage de conception/réalisation). Il doit aboutir à la concrétisation d'un projet bien défini au départ.

Mon travail s'inscrit dans le contexte de l'automatique et de l'informatique à la charge du service robotique, vision et réalité virtuelle (RV2) de l'INRIA Rhône-Alpes. Mon stage a pour but d'avancer la recherche sur un robot bipède, et plus particulièrement sur la connaissance de sa position dans l'espace à l'initialisation.

Dans ce document, je préciserai le cadre ainsi que le contexte du déroulement du début du travail de fin d'études, l'objectif à atteindre et la planification du travail.

Tout d'abord, je dresserai un portrait de l'INRIA Rhône-Alpes ainsi qu'une présentation du service robotique. Ensuite j'exposerai mes premiers résultats puis j'établirai un planning pour le déroulement de la fin du stage.

1 - Description de l'entreprise

1-1 PRESENTATION DE L'INRIA :

a) *Introduction :*

Créé en 1967 à Rocquencourt près de Paris, l'INRIA, Institut National de Recherche en Informatique et en Automatique, est un établissement public à caractère scientifique et technologique qui mène des recherches avancées dans le domaine des sciences et technologies de l'information et de la communication. Ce domaine inclut l'informatique et l'automatique, mais aussi les télécommunications et le multimédia, la robotique, le traitement du signal et le calcul scientifique. L'INRIA est placé sous la double tutelle du Ministère de la Recherche et du Ministère de l'Economie, des Finances et de l'Industrie.

L'INRIA a l'ambition d'être au plan mondial un institut de recherche au cœur de la société de l'information**.**

Sa volonté est de mettre en réseau des compétences et des talents de l'ensemble du dispositif de recherche français dans le domaine des STIC. Ce réseau permet de mettre l'excellence scientifique au service des progrès technologiques, créateurs d'emplois, de richesse et de nouveaux usages répondant à des besoins sociaux-économiques.

Son organisation décentralisée (5 unités de recherche), ses petites équipes autonomes et évaluées régulièrement permettent à l'INRIA d'amplifier ses partenariats, 47 projets de recherche sur 87 sont communs avec les universités, les grandes écoles et les organismes de recherche. Il renforce son implication dans les travaux de valorisation des résultats de recherche et le transfert technologique : 600 contrats R&D avec l'industrie et un peu moins d'une cinquantaine de sociétés sont issues de l'INRIA.

b) *Quelques chiffres (décembre 2000) :*

⇨ *Ressources budgétaires :* - dotation de l'état : 442 MF HT
- ressources propres : 174 MF HT

⇨ *Ressources humaines :*
- titulaires INRIA : 724.
- post-Doctorants et Contractuels: 256.
- doctorants : 550.
- chercheurs et enseignants d'autres organismes : 230.
- conseillers, collaborateurs divers et invités : 430.

⇨ *Indicateurs :*
- contrats de recettes actifs : plus de 600.
- contrats de recettes signés dans l'année : plus de 200.

- un peu moins d'une cinquantaine de sociétés sont issues de l'INRIA, depuis Ilog, aujourd'hui cotée au Nasdacq, jusqu'aux toutes dernières, 5 en 1998, 6 en 1999, 11 en 2000.
- 7 brevets initiaux déposés en 1999 : 1 est en pleine propriété INRIA, les autres sont en copropriétés, 5 avec des industriels et 1 avec une université.

c) *Gestion des projets :*

Un projet de recherche est une équipe rassemblant de 15 à 20 personnes autour d'une thématique forte et sur des objectifs scientifiques précis. Ces équipes gèrent de façon autonome leur budget. Les résultats obtenus et les retombées industrielles qu'ils induisent, sont régulièrement évalués.

Une action de recherche est un groupe de chercheurs poursuivant un travail commun sur une thématique spécifique qui peut aboutir à la création d'un projet de recherche.

1-2 PRESENTATION DE L'INRIA RHONE-ALPES :

a) *Introduction :*

Créée en décembre 1992, l'INRIA Rhône-Alpes est la plus récente des cinq unités de recherche de l'INRIA.

Menées au sein d'une région en plein essor technologique, les activités de l'unité de recherche INRIA Rhône-Alpes mobilisent plus de 330 personnes, dont 220 chercheurs, géographiquement réparties sur trois sites : le site de l'INRIA à Montbonnot, le campus universitaire de Grenoble et l'Ecole Normale Supérieure de Lyon.

Ces activités s'incrivent dans le cadre des missions que doit accomplir l'INRIA en tant qu'établissement de recherche national, tout en se focalisant sur les objectifs stratégiques poursuivis par l'institut dans le domaine des sciences et technologies de l'information et de la communication.

L'INRIA Rhône-Alpes accueille plus de 130 doctorants, ingénieurs et stagiaires. Ses chercheurs participent à l'enseignement supérieur au sein des universités et grandes écoles de la région Rhône-Alpes (Institut National Polytechnique de Grenoble, université Joseph Fourier, université Pierre Mendès-France, université de Savoie, Ecole Nationale Supérieure de Lyon).

b) *Missions de l'INRIA Rhône-Alpes :*

En tant qu'unité de recherche de l'INRIA, les principales missions de l'INRIA Rhône-Alpes sont, selon le décret du 2 août 1985 portant sur l'organisation et le fonctionnement de l'institut :

- ➪ *Entreprendre des recherches fondamentales et appliquées.*
- ➪ *Réaliser des systèmes expérimentaux.*
- ➪ *Organiser des échanges scientifiques internationaux.*

- ⇨ *Assurer le transfert et la diffusion des connaissances et du savoir-faire.*
- ⇨ *Contribuer à la valorisation des résultats de la recherche.*
- ⇨ *Contribuer, notamment par la formation, à des programmes de coopération pour le développement.*
- ⇨ *Effectuer des expertises scientifiques.*
- ⇨ *Contribuer à des actions de normalisation.*

c) ***Pôles de recherche :***

L'INRIA Rhône-Alpes mène ses activités en étroite collaboration avec les laboratoires de recherche publics et privés, nationaux et internationaux, et elle entretient des liens privilégiés avec l'institut d'Informatique et Mathématiques Appliquées de Grenoble (IMAG). Ces activités sont organisées autour de quatre pôles de recherche :

- ⇨ *Maîtriser les systèmes et réseaux informatiques* : Réseaux, parallélisme et systèmes répartis.
- ⇨ *Aider à la conception et à la création* : Bases de connaissances, documents multimédia, modèles cognitifs.
- ⇨ *Percevoir, simuler et agir* : Synthèse d'images, réalité virtuelle, vision par ordinateur et robotique.
- ⇨ *Modéliser les phénomènes complexes* : Automatique, simulation et calcul scientifique.

1-3 PRESENTATION DU SERVICE :

Mon stage se déroule au sein du service robotique, vision et réalité virtuelle (RV2) de l'INRIA Rhône-Alpes dont le rôle est la mise en oeuvre des outils matériels et logiciels pour les expérimentations robotiques des projets de recherche du site.

Ce service compte :

- ⇨ 3 ingénieurs de recherche,
- ⇨ 1 ingénieur expert,
- ⇨ 1 technicien,
- ⇨ 1 assistante de service.

a) ***Les missions du service :***

Les missions qui lui sont attribuées sont de trois types :

- ⇨ *Activité de service :*
  - ✓ maintenance des systèmes robotiques.
  - ✓ installation et maintenance de logiciels spécialisés.
  - ✓ interface entre les utilisateurs et le service informatique.
  - ✓ assistance aux utilisateurs.
- ⇨ *Activité de développement :*
  - ✓ mise en place d'expérimentations.
  - ✓ développement de logiciels dédiés à la robotique.
- ⇨ *Activité de recherche :*
  - ✓ conception de systèmes robotiques.
  - ✓ confrontation théorie et expérimentation.

Le but du Service Robotique est de fédérer l'effort expérimental en favorisant :

- ⇨ *les expérimentations inter-projets,*
- ⇨ *la mise en commun des moyens expérimentaux,*
- ⇨ *les outils réutilisables (environnement de développement, machine de vision...).*

b) *Les projets concernés :*

Les moyens robotiques travaillent avec les projets tels que "*Interaction homme-machine, images, données, connaissances*" et "*Simulation et optimisation de systèmes complexes*" impliqués en robotique et vision.

- ⇨ *SHARP* :Programmation automatique et systèmes décisionnels en robotique.
- ⇨ *MOVI* :Modélisation, localisation, reconnaissance et interprétation en vision par ordinateur.
- ⇨ *BIP* :Conception et contrôle de robots marcheurs et applications.
- ⇨ *IMAGIS* :Modèles, algorithmes, géométrie pour le graphique et l'image de synthèse.
- ⇨ *PRIMA* :développer des techniques pour l'intégration de la perception et de l'action en robotique.

2 - Présentation du stage

2-1 THEME DU STAGE : ROBOT BIPEDE

a) *Présentation :*

Le projet BIP, créé au 1er janvier 1994, a pour objectifs la conception de robots marcheurs de type bipède et, plus généralement, l'étude de techniques de contrôle/commande (algorithmes et architecture) de systèmes complexes.

L'intérêt de ces robots réside dans leur capacité naturelle à évoluer dans les environnements de notre vie quotidienne, essentiellement conçus pour la bipédie. Ainsi, la classe d'applications potentielles visée est-elle la robotique de service. Par ailleurs le robot actuellement utilisé dans l'équipe est conçu comme une plate-forme d'accueil pour des recherches en mécanique, automatique et informatique temps réel.

Le projet a pour objectif l'étude générique des divers aspects intervenant dans le contrôle/commande des systèmes robotiques complexes. Il s'intéresse en particulier aux robots marcheurs de type bipède. Les robots anthropomorphes sont particulièrement aptes à évoluer dans nos environnements courants, privés ou industriels, essentiellement conçus pour la bipédie. Ainsi, les domaines d'application visés sont-ils en priorité les robotiques personnelle, de service et d'intervention. Parallèlement, le projet s'attache à développer des activités de modélisation dans certains domaines de la biomécanique. Enfin, le projet ne souhaite pas que les applications de ses recherches se limitent au domaine des seuls robots marcheurs, dont le marché industriel est actuellement marginal. C'est pourquoi les techniques étudiées sont voulues suffisamment génériques pour faire l'objet de mises en oeuvre dans d'autres domaines.

L'intension n'est pas de reproduire la morphologie des jambes humaines qui est une machine extraordinaire comprenant quelques 55 os, 90 muscles et 16 nerfs dont une réalisation mécanique précise peut largement dépasser la technologie courante. Le but est plutôt d'appliquer le comportement anthropormorphe à une version simplifiée des jambes.

Le robot, en étude depuis cinq ans dans le cadre d'un projet plurilaboratoire, comporte deux jambes et un tronc, et possèdera 17 articulations dans sa version finale. Ses dimensions et sa structure d'actionnement des jambes sont inspirées de la cinématique humaine. Sans équivalent à notre connaissance en Europe, ce système représente la synthèse de recherches avancées en conception de mécanismes, en automatique et en informatique.

b) *Acteurs du projet :*

Un projet d'une telle envergure demande la collaboration de plusieurs laboratoires :

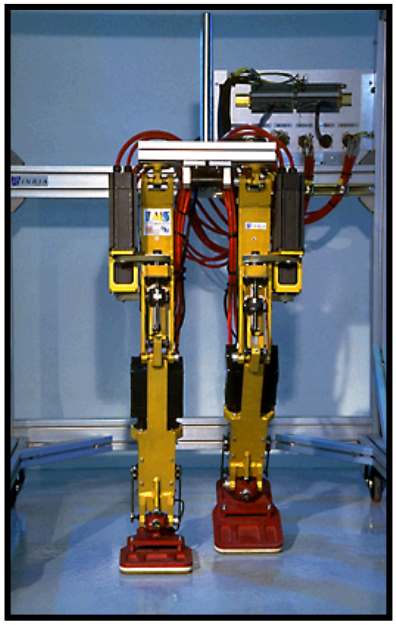
➪ *Projet BIP de l 'INRIA Rhône-Alpes* : responsable de la partie contrôleur.
➪ *Laboratoire de Mécanique des Solides de Poitiers* : responsable de la partie mécanique.
➪ *Services Moyens Robotiques Vision et Réalité Virtuelle* : chargé de la partie électronique.

c) *Axes de recherche :*

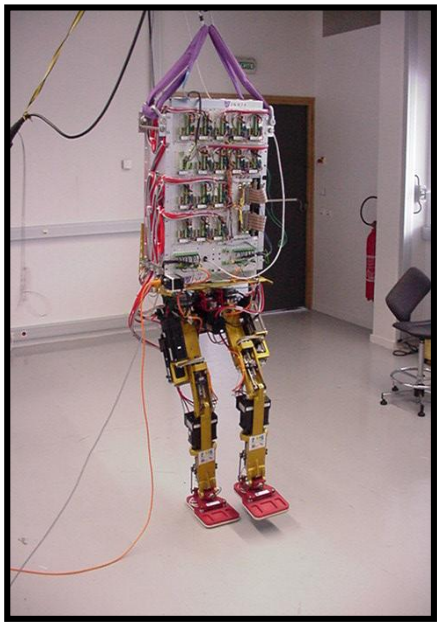
➪ Modélisation de la marche humaine dans diverses configurations (en collaboration avec des biomécaniciens) : mesures de paramètres sur des ensembles de sujets, recherche d'invariants posturaux ou de mouvement, détermination des schémas de contrôle sous-jacents.
➪ Etude de méthodes de commande basées sur les régimes passifs quasi-périodiques, la stabilisation d'ensembles, les tâches référencées capteurs (force, proximité ou vision).
➪ Mise au point d'outils de conception, programmation et vérification pour l'ensemble du contrôle/commande à partir d'une approche synchrone, études de tolérance aux fautes.

d) *Evolution :*

Voici quelques photos du robot qui retranscrivent son évolution (le passage de 8 a 15 DDLs) :



Le premier prototype(BIP1) de robot bipède à 8 degrés de liberté. BIP1 a fait ses premiers pas en mars 2000.



Robot bipède à 15 articulations (BIP2) capable de maintenir son équilibre statique sur un pied tout en bougeant l'autre en l'air.

2-2 PROBLEMATIQUE SCIENTIFIQUE :

La difficulté de conception d'un robot marcheur bipède résulte, d'une part de la nouveauté du concept, qui fait qu'il n'existe pas encore de solutions éprouvées, et d'autre part de contraintes spécifiques au principe même de la marche bipédique : tout d'abord, le système doit comporter un nombre élevé d'articulations dans un espace de volume réduit. De plus, la localisation de celles-ci est elle-même fort contrainte par des raisons de position des articulations, d'encombrement et de répartition des masses, ce qui nécessite par exemple l'étude de groupes moto-transmetteurs spécifiques. Par ailleurs, le système doit être en permanence en équilibre, soit statique, soit dynamique, ce qui peut nécessiter des couples instantanés élevés. Ceux-ci se répartissent d'ailleurs au cours du temps sur l'ensemble des articulations, posant ainsi le problème d'une gestion globale et dynamique de la puissance nécessaire. Ce dernier point est également lié aux choix technologiques concernant l'autonomie : la place disponible est faible et la masse embarquable limitée (elle se situe en général en hauteur (tronc) et a de ce fait une grande influence sur la dynamique du système).

2-3 L'ETAT D'AVANCEMENT DU PROJET EST LE SUIVANT :

➪ La première version du robot BIP2000, appelé BIP1, possède deux jambes avec 8 degrés de liberté. BIP1 a fait ses premiers pas en mars 2000.

➪ La seconde version, appelée BIP2, possède deux jambes, un bassin articulé et un tronc, avec 15 degrés de liberté. BIP2 est capable de maintenir son équilibre statique sur un pied tout en bougeant l'autre pied en l'air.

➪ L'environnement de programmation pour le contrôle-commande temps-réel du système est opérationnel. Il s'agit d'ORCCAD.

2-4 L'ETAT ACTUEL :

L'objectif initial du projet était la réalisation d'un bipède à 17 degrés de liberté, aux dimensions et aux masses des membres inférieurs voisines de celles de l'homme. Dans sa première version, le système ne devait porter qu'un tronc intégrant l'électronique de commande, donc sans bras ni tête. Il n'était pas non plus censé être autonome sur le plan énergétique. Il devait être capable de marcher de façon anthropomorphe en l'absence d'obstacles sur sol horizontal ou légèrement incliné, et de monter ou descendre des escaliers. Ce cahier des charges est maintenu pour BIP 2000, disposant de 15 degrés de liberté avec 3 mobilités opérationnelles au niveau du tronc. Depuis le démarrage effectif du projet en 1994, les recherches ont progressé sur deux fronts :

➪ *Les études amont :*

Une analyse expérimentale de la marche humaine sur plans inclinés et pendant la montée ou la descente d'escaliers a été réalisée, mettant en évidence quelques paramètres significatifs à intégrer dans la commande du robot. La modélisation et la recherche des cycles naturels des systèmes mécaniques avec impact a également avancé et des lois de commande, simples ou optimales ont été proposées. La conception de la mécanique a fait l'objet d'études d'optimisation et un nouveau système transmetteur a été conçu.

➪ *Les réalisations :*

Un environnement logiciel complet permettant la spécification, la vérification formelle et la programmation temps réel automatique du contrôle/commande a été développé. Une première version de jambe en alliage d'aluminium a été réalisée. Grâce à celle-ci, le test de l'articulation complète du genou a été effectuée. Un robot à 8 dégrés de liberté, composé de deux jambes est maintenant opérationnel. Son jumeau à 15 degrés de liberté, qui dispose de deux jambes, d'un pelvis et d'un tronc, a été testé avec des trajectoires 3D en avril 2000.

2-5 SUJET DU STAGE :

Le principal but de mon stage est de programmer des tâches élémentaires sur le robot BIP2000. Une tâche élémentaire peut être par exemple "se mettre dans une position donnée à la mise sous tension".

Ce stage permet de suivre le projet de la conception à l'implémentation, d'intégrer des composants matériels et logiciels complexes.

3- Méthodologie

3-1 PRESENTATION DES ORGANES DU ROBOT INTERVENANT DANS LE POSITIONNEMENT :

a) *Les actionneurs :*

Les actionneurs utilisés sont des servomoteurs PARVEX LX avec servoamplificateurs SBS. Les moteurs PARVEX peuvent être au choix commandés en courant ou en vitesse(le choix se fait par l'intermediaire d'un « switch » physique). Pour différentes raisons (-temps de réponse de la commande en vitesse est beaucoup plus long que pour la commande en courant, - la commande en vitesse n'est satisfaisante que pour des vitesses constantes, ce qui est rarement le cas du robot), le choix d'une commande en courant est plus adapté.

Les axes moteurs sont équipés d'un système de compteur optique de position. Un tour est régulièrement subdivisé en un nombre entier de tops codeurs. Un codeur incrémental permet à chaque instant de connaître la position du moteur par rapport à la position initiale. Le compteur est mis à zéro à l 'initialisation quelque soit la position réelle du robot.

b) *Les potentiomètres :*

Pourquoi ce choix ?

Lors de l'initialisation du robot, les codeurs de position sur les axes moteurs sont mis à zéro. Ces codeurs permettent donc de connaître après un déplacement, les angles moteurs relatifs par rapport à ce zéro. Il a donc fallu trouver une solution pour pouvoir déterminer la position absolue du robot.

➪ Une solution envisagée était de bouger les membres du robot afin de venir buter physiquement sur une position de référence, mais l'inconvénient majeur était le risque de perte d'équilibre et la chute du robot.

➪ La seconde solution a été de rajouter des détecteurs de position absolue sur chaque articulation. Cette solution a été retenue pour la détermination de la position absolue du robot. Les détecteurs choisis sont des potentiomètres. Ils donnent une tension proportionnelle à l 'angle formé par l 'articulation. Leur précision est largement inférieur à celle des capteurs optiques au niveau moteur. Ils ne seront donc utilises qu'une seule fois, c'est-à-dire à l'initialisation pour connaître la position absolue du robot. Les codeurs incrémentaux prennent ensuite le relais pour connaître la position relative.

Cette technologie de potentiomètre a été choisie pour des raisons de cout et également d'encombrement.

Un potentiomètre coûte environs 250 francs alors qu'un codeur absolu (autre solution envisagée) coûte 10 fois plus chère.

Sur chaque axe, un potentiomètre a été fixé, ce qui a engendré quelques modifications :

- ⇨ Problèmes mécaniques pour leur fixation.
- ⇨ Ajout d'un module de conversion analogique/numérique supplémentaire.
- ⇨ Phase de calibration.

Les différents potentiomètres :

Deux types de potentiomètres sont utilisés suivant la place disponible du point de vue mécanique.

Il s'agit de potentiomètres monotours sans butée et dédiés à ce type d'utilisation. L'angle peut être connu par la lecture de la tension du point milieu. La tension d'alimentation est imposée par les convertisseurs analogique/numérique dont l'entrée est comprise entre –10V et +10V. Sachant que les tensions disponibles a partir du robot sont +5V et +15V, la tension est obtenue par un diviseur potentiomètrique. (cf. fig ci-dessous)

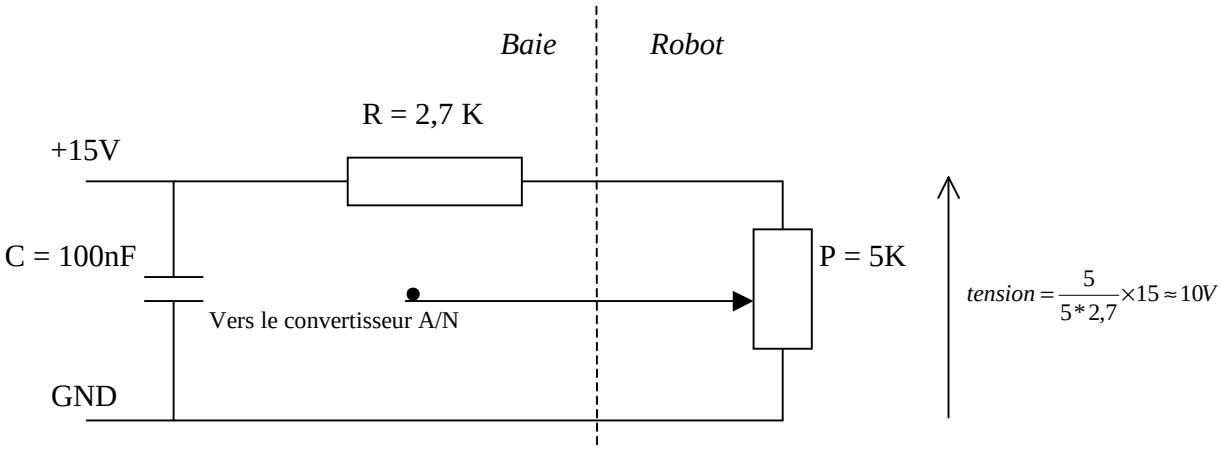


Schéma de cablage des potentiomètres

- Le premier modèle MEGATRON se fixe par collage de l'axe du potentiomètre dans un trou situé sur l'axe de rotation de chaque articulation.
Les données constructeur pour ce modèle sont les suivantes :
  - ⇨ Résistance totale : 5kΩ.
  - ⇨ Angle électrique : 340 °.
  - ⇨ Résolution : 0,01 °.
  - ⇨ Linéarité standard indépendante de ± 0,25%.

- Le second modèle VISHAY a été spécifié et fabriqué en sous-traitance. Ces potentiomètres se composent de 2 parties, un curseur (fixé sur la partie dite mobile) et un support pour la piste résistive (fixé sur l'autre partie de l'articulation dite fixe). Les valeurs spécifiques sont les suivantes :
  - ⇨ Résistance totale : 5kΩ.
  - ⇨ Angle électrique : 45 °.

Chaque axe du robot est équipé d'un potentiomètre pour la lecture de sa position angulaire absolue. L'information est disponible sous forme analogique, elle doit donc être convertie via un module IP_ADC_potar pour être exploitable par le calculateur embarqué.

Le calculateur du robot est basé sur une carte Motorola MVME162 supportant le système d'exploitation temps-réel VxWorks.

3-2 ARCHITECTURE INFORMATIQUE :

Le tronc du robot contient une armoire qui comporte l'électronique de puissance et de commande. La commande temps-réel du robot se fait sous le système d'exploitation VxWorks et l'environnement Orccad (développé par l'INRIA) depuis une station de travail. Le robot est relié par un cordon ombilical à une source de tension et à la station de travail via un lien ethernet.

3-3 PRESENTATION DE VXWORKS :

VxWorks est un système d'exploitation temps-réel dans lequel les vérifications des accès mémoire ont été supprimées afin d'accroître les capacités temps réel. Il nécessiterait un chapitre à lui tout seul, mais n'étant pas le but du travail mais seulement une plateforme de développement, je n'en présenterais que les grandes lignes.

C'est le premier système à intégrer le cross-développement (développement pour une autre plateforme ou développement croisé) à partir d'une station unix. Il adhère à la norme POSIX et permet de programmer en C, C++, Java, etc...Enfin, il supporte les bus VME et PCI.

La mémoire est partagée entre toutes les tâches, la table des symboles est globale, et il n'y a pas de protection en écriture.

Toutes les fonctionnalités sont implémentées sous forme de bibliothèques de fonctions C. Une fonction C peut être :

- appelée interactivement depuis le shell
- lancée comme une tâche VxWorks
- connectée à une interruption ou à un timer (horloge)

De plus, une bibliothèque peut-être chargée soit statiquement à la construction du noyau, soit avec le chargeur (loader) dynamique de fonctions depuis le shell ou depuis une fonction C.

Les modules sont chargés un par un et démarrés en lançant les fonctions exportées qui s'y trouvent.

Le developpement logiciel du code embarqué se fait donc au travers des outils WIND RIVER Systems, c'est-à-dire TORNADO. Les programmes sont d'abord compilés de façon croisée sur station SUN solaris puis une commande de VxWorks télécharge le code sur la carte cible MVME162.

3-4 PROCEDURE D'ETALONNAGE DES POTENTIOMETRES :

⇨ **Installation définitive des potentiomètres** avec de la colle (Plusieurs essais ont permis de choisir la colle appropriée à base de résine époxy).

⇨ **Vérification de la concordance entre les valeurs des potentiomètres du point de vue software et hardware**. (la valeur hardware est rélevée avec un voltmètre, et la valeur software est donnée à l'aide d'un programme de test, un menu permet d'accéder aux valeurs des potentiomètres).
(cf. annexe 1 : tableau de valeurs)

⇨ **Réalisation d 'une méthodologie pour la mise en position au zéro mécanique**. Tous les calculs effectués auparavant pour donner la position angulaire de chaque articulation en fonction de la valeur codeur ont été réalisés par rapport à une position initiale bien précise. Le robot ayant été démonté, il a fallu le repositionner dans cette position souhaitée. Connaissant un positionnement théorique, il a fallu mettre en place une méthodologie réutilisable à chaque fois que le robot pourra être démonté.
(cf. annexe 2 : Présentation de la méthodologie)

⇨ **Réalisation d 'un programme relevant à la fois les tensions des potentiomètres et les codeurs** (qui sont ensuite convertis en position angulaire).
(cf. annexe 3 : Code du programme)
Programme utilisant les fonctions :

int bipHardInit() : Init the hardware of bip.

*int bipHardGetAnalog(int channel, double *value)* : Get the value on ADC channel, the value given is in [-10.0...10.0].

*int bipHardGetEncoders(int channel, double *value)* : Get the values of the quadrature channels (encoder motors), channel = 0...14.

int bipUtilCalcPos(double th[BIP_DDL], double q[BIP_DDL]) : traduit les positions moteurs th en positions articulaires q.

⇨ **Réalisation de l'expérimentation associée à ce programme** : il s'agit d'enregistrer les deux valeurs (potentiomètre et codeur) pour des positions différentes de l'articulation. On effectue donc plusieurs aller-retour de chaque articulation et on relève les valeurs.
Attention : nécessité du positionnement en zéro mécanique lors de l'initialisation (pour que les valeurs codeurs aient une signification).
(cf. annexe 4 : tableau récapitulatif des paramètres de l'expérimentation : durée et fréquence de mesures)

⇨ **Exploitation des données avec Maple**. Tracé de la courbe : position angulaire (en utilisant les codeurs) en fonction de la tension du potentiomètre. Cette courbe donne la calibration des potentiomètres. On obtient ainsi un offset et une pente pour chaque articulation.
(cf. annexe 5 : Exemple de résultat obtenu avec maple pour une articulation)

⇨ **Réalisation d'une fonction permettant d 'obtenir la position angulaire en utilisant les potentiomètres**.
(cf. annexe 6 : Code du programme)

⇨ **Vérification des résultats obtenus** : on se place en position 0 (une des seules positions connues) et on analyse les résultats donnés par le programme précédent. On peut ainsi estimer l'erreur.

4 - Conclusion

4-1 ANALYSE DES RESULTATS OBTENUS :

La linéarité des potentiomètres n'est pas parfaite, et il subsiste un phénomène d'hystérésis. Précédemment, nous avions estimé une variation de $2*10^{-2}$ pour la mesure software des potentiomètres (valeur estimée en répétant plusieurs fois la mesure software sans modifier la position du robot). Compte tenu des pentes obtenues par la méthode des moindres carrés (avec maple), cette valeur convertie en position angulaire donne une erreur de l'ordre de 0,6°.

L'hystérésis mécanique (jeu entre les axes, efficacité du collage) et l' hystérésis électrique (changement de sens de rotation du curseur) sont également des causes d'erreur.

Une étude a été réalisée et a montré qu'¼ de degrés sur chaque articulation correspond à une erreur de 4cm sur la position du bout du pied en l'air lors de l'appui unipostal. Plus généralement, une erreur d'1 degrés sur un angle articulaire correspond à une erreur de position de 17 mm un mètre plus loin.

La précision souhaitée pour valider l'utilisation des potentiomètres était de ¼ de degrés. En effectuant une valeur moyennée sur 50 mesures (pour une même position du robot, nous relevons 50 valeurs de la tension puis nous en faisons une moyenne), nous obtenons une erreur de 0,2°. Cette valeur garantit donc nos attentes.

4-2 PERSPECTIVES :

Mois	***Perspectives***
Juin	⇨ Utiliser le réseau pour connaître l'état du robot. ⇨ Etude du modèle.
Juillet	⇨ Réglage des PID. (Problème du double appui) ⇨ Améliorer l'initialisation avec les tops zéro des moteurs.
Août	⇨ Exécuter les démos existantes (effectuée sur le robot du *Laboratoire de Mécanique des Solides de Poitiers*).
Septembre	⇨ Mise sous tension, position de départ. ⇨ Marche statique (suivi de trajectoires). ⇨ Mise au propre des documents et des programmes.

ANNEXE 1 : MESURE DES VALEURS DES POTARS POUR LA COMPARAISON DES VALEURS HARDWARE ET SOFTWARE :

Précision des mesures :

➪ pour les valeurs hardware (valeurs relevées au point milieu des potentiomètres à l'aide d'un voltmètre), la précision est jugée à 10^{-2} volts près.

➪ pour les valeurs sofware, la précision est de l'ordre de $2*10^{-2}$ près, cette précision est estimée ainsi, du fait que pour une même position du robot, si l'on effectue plusieurs fois l'acquisition software des tensions, on trouve un écart entre les valeurs extêmes de $2*10^{-2}$.

				mesures en tension de la partie hardware et software des potars								
n° articulation	*Articulation*	*Situation*	mesures	n°1	n°2	n°3	n°4	n°5	n°6	n°7	n°8	tension U
0	Cheville axe frontal	J.D.	*tension hardware*	6,24	5,95	5,51	5,07					9,20
			tension software	6,237	5,956	5,507	5,065					
1	Cheville axe sagittal	J.D.	*tension hardware*	5,31	4,71	4,32	3,57	3,1				9,20
			tension software	5,307	4,702	4,316	3,576	3,101				
2	Genou axe sagittal	J.D.	*tension hardware*	5,82	5,33	4,69	4,2	3,3				9,04
			tension software	5,825	5,326	4,69	4,197	3,295				
3	Hanche axe sagittal	J.D.	*tension hardware*	6,31	6,15	6,02	5,68	5,49				9,37
			tension software	6,298	6,146	6,021	5,682	5,495				
4	Cheville axe frontal	J.G.	*tension hardware*	6,29	5,78	5,31	5,13					9,39
			tension software	6,298	5,78	5,312	5,127					
5	Cheville axe sagittal	J.G.	*tension hardware*	4,28	4,91	5,45	6,85					9,40
			tension software	4,284	4,901	5,442	6,849					
6	Genou axe sagittal	J.G.	*tension hardware*	3,59	3,93	4,26	4,62	5,24	6,21			9,23
			tension software	3,589	3,914	4,267	4,615	5,255	6,203			
7	Hanche axe sagittal	J.G.	*tension hardware*	3,48	3,69	3,88	4,20	4,42	4,72			9,31
			tension software	3,479	3,691	3,875	4,195	4,424	4,709			
8	Hanche axe vertical	J.D.	*tension hardware*	5,25	4,70	3,61	2,53	1,56				9,10
			tension software	5,256	4,708	3,604	2,527	1,564				
9	Hanche axe frontal	J.D.	*tension hardware*	0,71	1,93	3,87	5,94	7,21				9,04
			tension software	0,705	1,928	3,872	5,927	7,208				
10	Hanche axe vertical	J.G.	*tension hardware*	3,22	4,11	4,76	5,47	6,18	6,90	7,54		9,05
			tension software	3,214	4,124	4,763	5,463	6,179	6,906	7,530		
11	Hanche axe frontal	J.G.	*tension hardware*	7,17	5,61	4,54	3,45	2,36	0,89	0,75		8,92
			tension software	7,157	5,612	4,527	3,444	2,366	0,888	0,746		
12	Tronc axe vertical	Bassin	*tension hardware*	2,02	2,46	3,27	3,69	4,27	4,85	5,86	6,93	9,03
			tension software	2,017	2,459	3,272	3,683	4,259	4,856	5,856	6,924	
13	Tronc axe frontal	Lombaire	*tension hardware*	2,11	2,16	2,24	2,33	2,40				9,29
			tension hardware	2,106	2,164	2,237	2,323	2,401				
14	Tronc axe sagittal	Lombaire	*tension hardware*	8,98	8,81	8,63	8,46	8,24				9,26
			tension software	8,966	8,811	8,618	8,465	8,236				

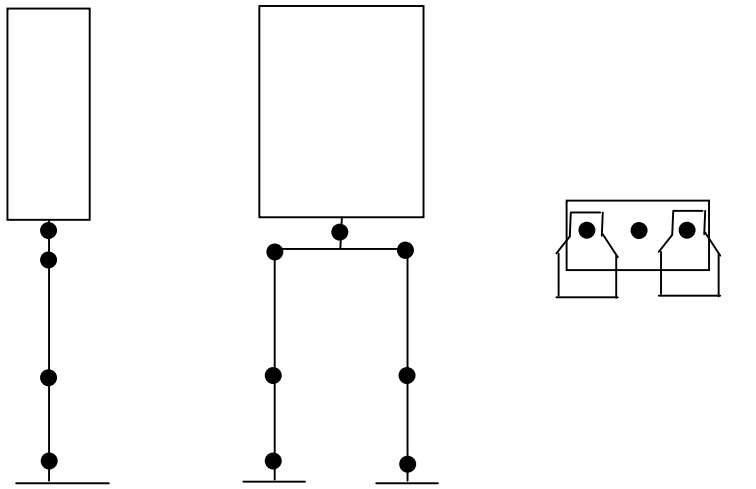
J.D. : Jambe Droite
J.G. : Jambe Gauche
tension U : tension aux bornes extremes du potentiomètre

ANNEXE 2 : METHODOLOGIE POUR LA MISE EN POSITION AU ZERO MECANIQUE (INITIALISATION) DU ROBOT :

Le document suivant présente la méthodologie à suivre pour réaliser la mise en **position zéro** du robot. Certaines phases de réglage sont redondantes, mais permettent une mise en position précise et rigoureuse du robot.

Pour que cette mise en position soit la simple possible, il est préférable de **suivre les étapes dans l'ordre** car certains réglages sont dépendants des autres (un mauvais réglage d'une articulation peut engendrer la nécessité du repositionnement de plusieurs autres degrés de liberté).

Position zéro souhaitée :



Vue de coté Vue de face Vue de dessus

Mise en position de la jambe droite / de la jambe gauche et du tronc :

Matériel utilisé :

- 1 réglet.
- 1 équerre.
- 1 niveau en guise de plan.
- 2 inclinomètres de préférence calibrés de la même façon.
- 1 oscilloscope ou 1 voltmètre.

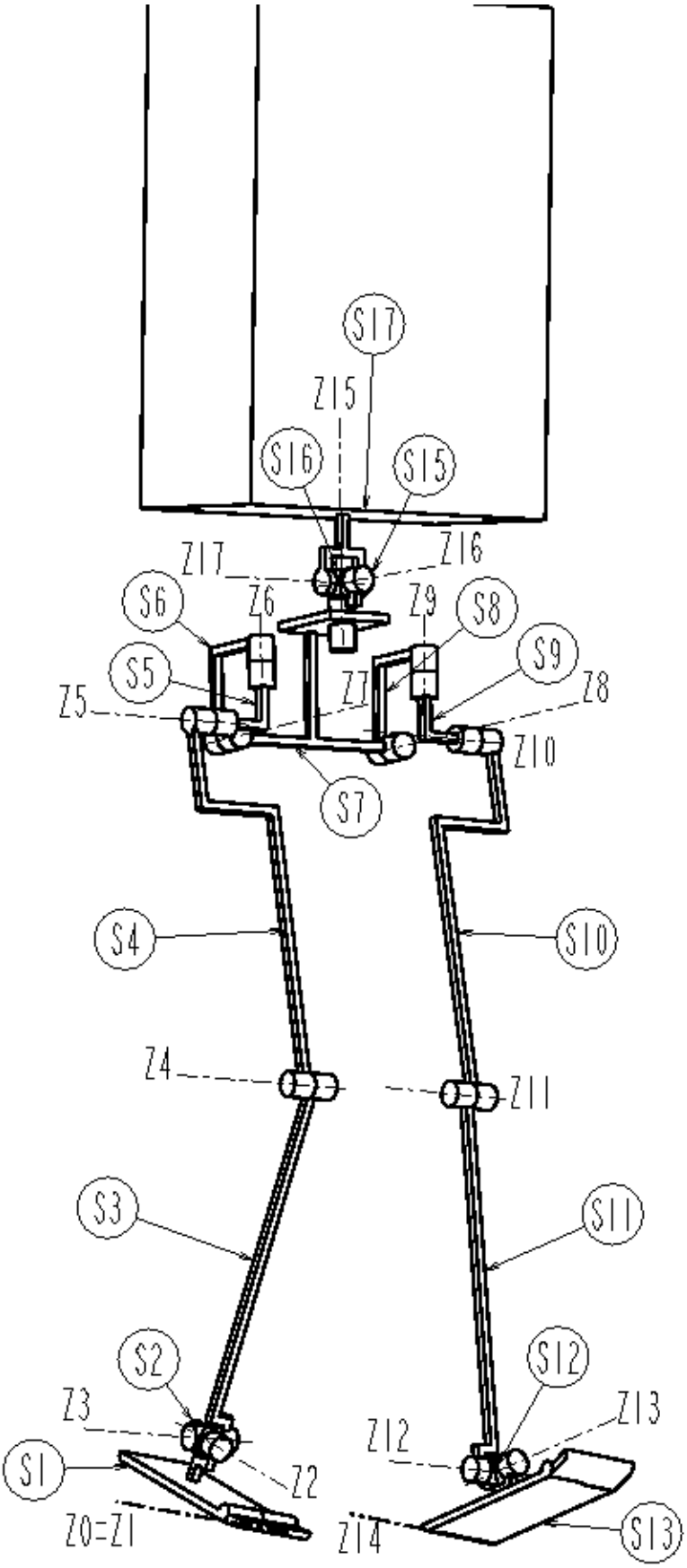
Précautions à prendre :

A la fin de chaque réglage, il est préférable de **vérifier les articulations précédemment positionnées**, notamment pour le genou où il réside un léger jeu.

Dans un premier temps, il est intéressant de réaliser une mise à zéro approximative. Il s'agit de régler chaque articulation en mesurant **la longueur de vis dépassant des écrous** (les valeurs suivantes sont valables seulement si les moteurs n'ont pas été démontés)

- pour les hanches : entre 58 et 59 mm
- pour les lombaires : entre 72 et 73 mm
- pour les chevilles : entre 53 et 54 mm.

SCHEMA DU ROBOT



Mise en position du genou :

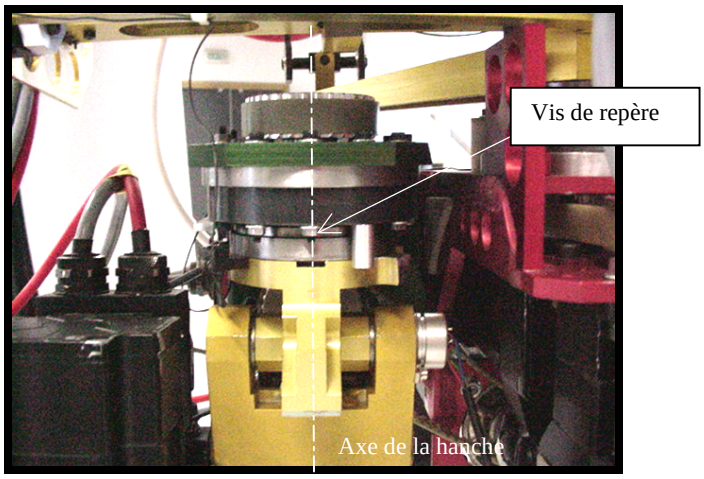
Axe : Z4 / Z11
Moteur associé : 2 / 6
Méthodologie :

- Mettre le genou en butée mécanique jambe tendue. (cette articulation est à vérifier fréquemment car il réside un léger jeu qui peut modifier les positions de certaines autres articulations)

Mise en position de la hanche axe frontal :

Axe : Z6 / Z9
Moteur associé : 9 / 11
Méthodologie :

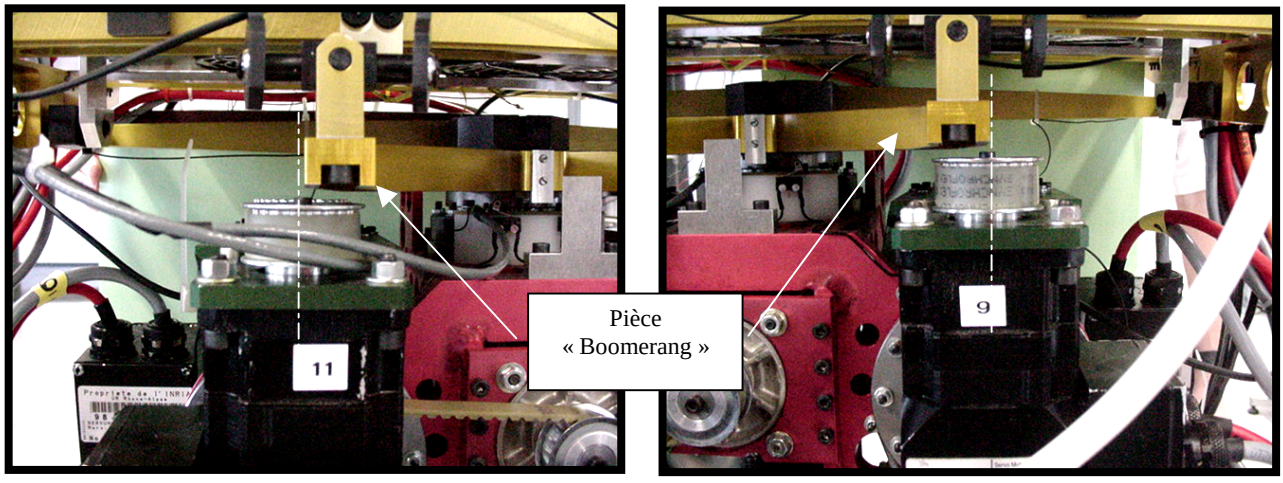
- Faire correspondre le mieux possible l'axe de la vis avec l'axe de la hanche.



Mise en position du tronc axe vertical :

Axe : Z16
Moteur associé : 12
Méthodologie :

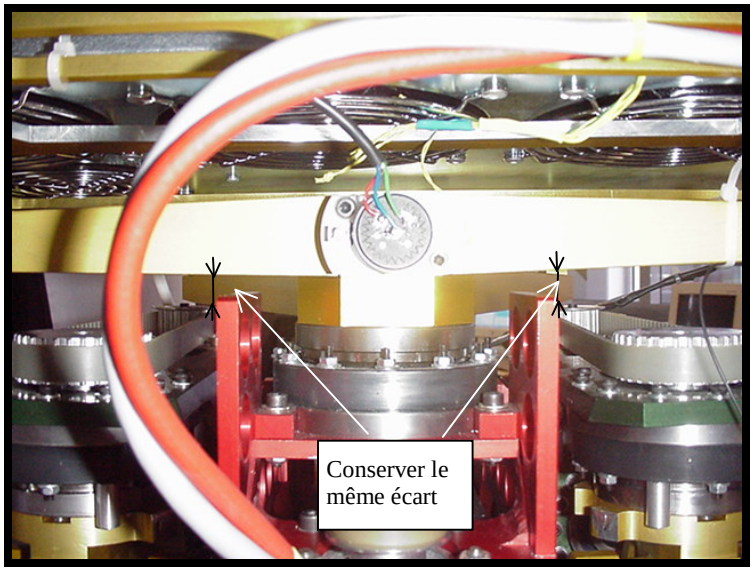
- Centrer visuellement le boomerang sur l'axe des moteurs 9 et 11.



Mise en position du tronc axe frontal :

Axe : Z16
Moteur associé : 13
Méthodologie :

- Mesurer les écartements (avec un réglet) représentés sur la photo ci-dessous.
- Ces mesures doivent être égales (Pour faciliter plus tard la mise en position sagittal du tronc, approcher cette valeur à 7 mm).



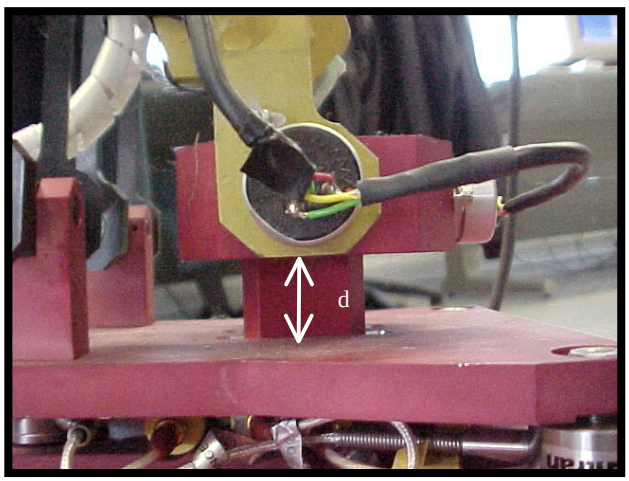
Mise en position de la cheville axe frontal et axe sagittal :

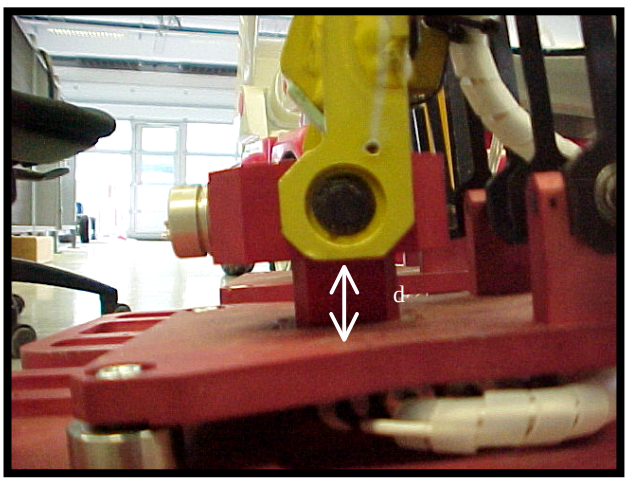
Axe (frontal) : Z2 / Z13
Moteur associé : 0 / 4

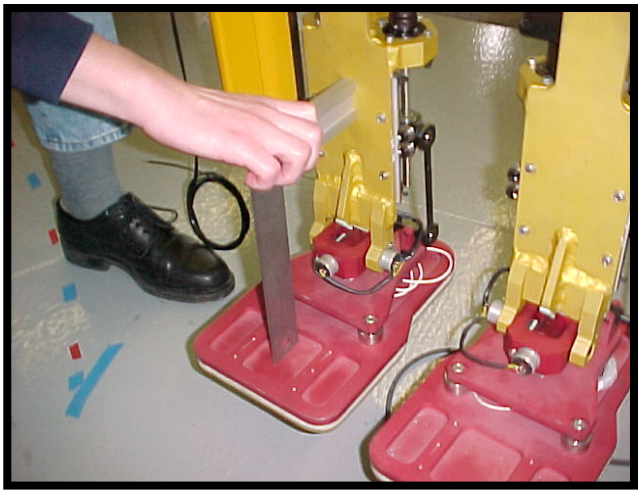
Axe(sagittal) : Z3 / Z12
Moteur associé : 1 / 5

Méthodologie :

- Sur chaque coté de chaque cheville, mesurer la distance d (cf. photo).Cette valeur doit être égale à 2 cm. Cette étape est très importante pour la suite, car elle permet à la fois la perpendicularité de la cheville dans l'axe sagittal et dans l'axe frontal.



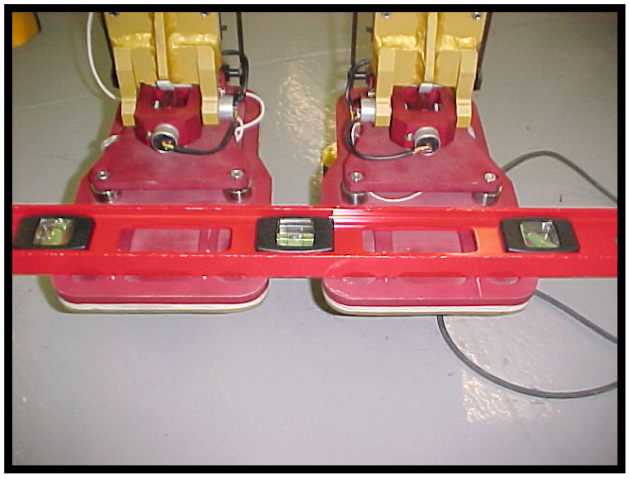




- S'assurer avec l'équerre de la perpendicularité du pieds par rapport au mollet. (cf. photo ci-contre)

Mise en position de la hanche axe vertical :

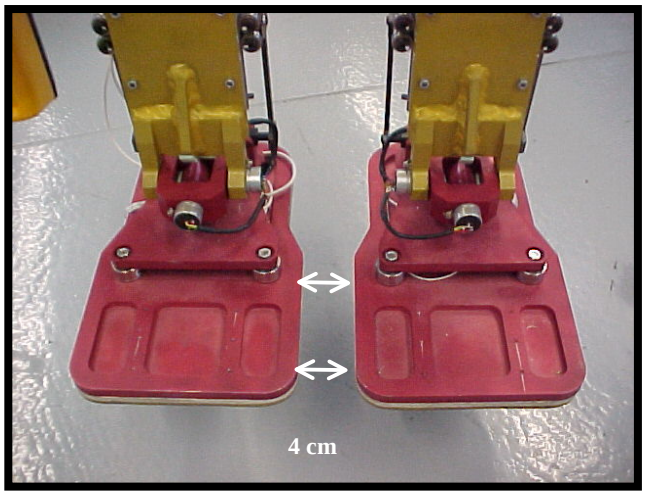
Axe : Z7 / Z8
Moteur associé : 8 / 10
Méthodologie :



- Cette phase est à effectuer une fois la perpendicularité des 2 pieds vérifiée. On vérifie ainsi que les 2 pieds sont situés sur un même plan. On doit dans un premier temps mettre les hanches dans une même position dans l'axe sagittal (la position zéro n'est pas obligatoire).

- S'assurer que le plan soit totalement en contact avec les 2 pieds. Pour corriger l'erreur, il faut agir sur les positions des hanches dans l'axe vertical.

- Pour vérifier que les 2 pieds soient bien situés dans une même direction, on peut utiliser 2 inclinomètres, un posé sur chaque pied. Si les 2 inclinomètres (calibrés de la même façon) donnent un signal identique, cela signifie que les 2 pieds sont bien positionnés dans la même direction, mais pas forcément dans le même plan.



- Une fois les 2 pieds bien positionnés (axe frontal et axe sagittal), on mesure l'écart entre eux (cf. photo ci-contre). L'écart doit être égal à 4 cm. Si cette mesure n'est pas correcte il faut repartir du positionnement des chevilles.

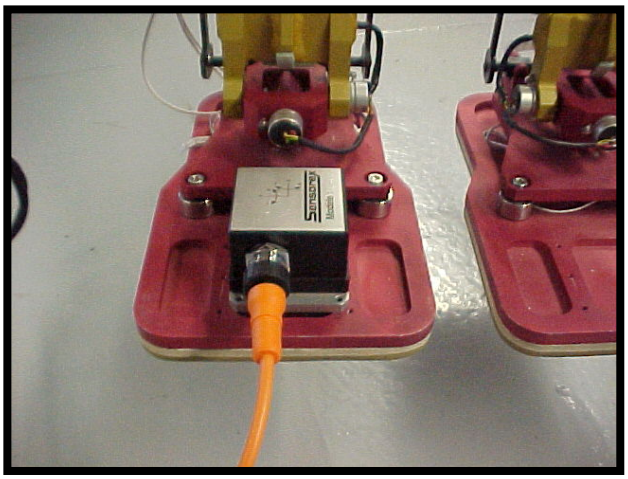
- Cette phase doit consister en une vérification, si les étapes précédentes ont été convenablement effectuées.

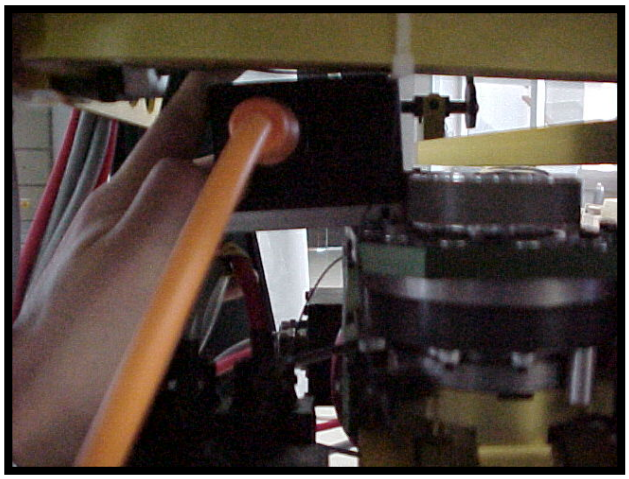
Si toutes ces conditions (frontale et axiale) sont satisfaites, on garantit la mise en position 0 de la cheville.

Mise en position de la hanche axe sagittal :

Axe : Z5 / Z10
Moteur associé : 3 / 7
Méthodologie :

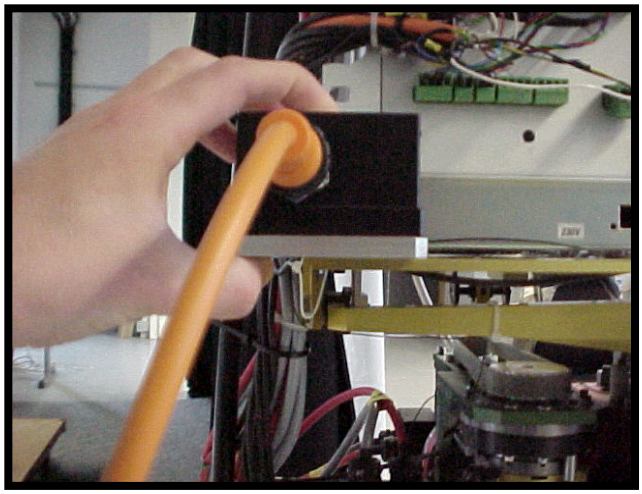
- On utilise deux inclinomètres : le premier posé sur le pied (celui étant déjà positionné correctement) et le second sur la surface désignée sur la photo ci-dessous. On désire le parallélisme de ces 2 plans, on peut donc travailler en position relative, c'est-à-dire qu'il nous faut un signal identique sur les 2 inclinomètres (ceux-ci étant calibrés de la même façon).





Mise en position du tronc axe sagittal :

Axe : Z17
Moteur associé : 14
Méthodologie :



- On souhaite ici le parallélisme entre le tronc et les pieds (par exemple). On utilise également 2 inclinomètres, un posé sur un des 2 pieds et l'autre posé sur la surface indiquée ci-contre. (on travaille également en position relative donc on doit positionner le tronc pour obtenir le même signal sur les 2 inclinomètres).

Etape de repérage :

Une fois tous ces réglages réalisés, il est souhaitable de repérer la position des moteurs en marquant à la fois le bati et le moteur (avec du tipex). Ainsi, il sera possible de positionner relativement facilement le robot en position zéro en faisant correspondre les marques.

ANNEXE 3 : REALISATION D 'UN PROGRAMME RELEVANT A LA FOIS LES TENSIONS DES POTENTIOMETRES ET LES CODEURS

```
#include <stdio.h>
#include <math.h>

#ifdef VXWORKS
#include <taskLib.h>
#include <tickLib.h>
#include <sysLib.h>
#include <semLib.h>
#endif

#include "hardBip.h"
#include "utilBip15.h"

#define CHEMIN "/home/ciney/bozonnet/Mesures"

/*-----------------------------------------------------*/
/*  Definition des variables et des fonctions   */
/*-----------------------------------------------------*/
#define BIP_DDL 15
#define DIM_TABLEAU 5000

static int bipLectSaveFile(char *name, double tableau[DIM_TABLEAU][BIP_DDL]);

static SEM_ID SemTache;
static SEM_ID SemTmp;

static int bipLectOverrun;
static int bipLectOverrunFlag;
static int bipLectExperimentation; /* variable attribuee a la fin de l'experimentation*/

static double bipLectCodeur[BIP_DDL];
static double bipLectArt[DIM_TABLEAU][BIP_DDL];
static double bipLectAdc[DIM_TABLEAU][BIP_DDL];

static int bipLectInitFlag = ERROR; /*variable gerant l'initialisation*/

/*CODEUR:tableau de passage des entrees module_ip aux entrees bipede*/
static int bipLectCodeurCor[BIP_DDL]={12, 13, 14, 15, 4, 5, 6, 7, 8, 9, 0, 1, 2, 10 ,11};

/*POTAR:tableau de passage des entrees module_ip aux entrees bipede*/
static int bipLectPotarCor[BIP_DDL] = {3, 11, 12, 4, 1, 9, 10, 2, 7, 6, 5, 13, 14, 8, 16};

static int bipLect      /* variable correspondant a l'axe desire */
static int bipLectCpt;



/*-----------------------------------------------------*/
/*  fonction de travail: lecture des donnees   */
/*       a un echantillonnage choisi                */
```

```
/*-----------------------------------------------*/
static int bipPeriodTache()
{
  int c;

  while(bipLectExperimentation == 0)     /* test sur la fin de l'experimentation */
    {
      semTake(SemTache, WAIT_FOREVER);
      if (bipLectOverrunFlag == 1)
            {
              fprintf(stderr, "bipLectPeriod: overrun detected\n");
              return(ERROR);
            }

      bipLectOverrun = 1;

      /* LIRE LES DONNEES EN ENTREE */
      semTake(SemTmp, WAIT_FOREVER);
      for (c=0;c<BIP_DDL;c++)
            {
              bipHardGetEncoders(bipLectCodeurCor[c],
                                 &bipLectCodeur[c]);
            }

      if (bipUtilCalcPos(bipLectCodeur, bipLectArt[bipLectCpt]) ==ERROR)
            {
              printf("bipUtilCalcPos ERROR\n");
            }

      semGive(SemTmp);

      semTake(SemTmp, WAIT_FOREVER);

      bipHardGetAnalog(bipLectPotarCor[bipLectAxe]+15,
                       &bipLectAdc[bipLectCpt][bipLectAxe]);
      if (bipLectCpt < DIM_TABLEAU)
            {
              bipLectCpt++;
            }
      else
            {
              printf("bipLectCpt: depassement\n");
            }

      semGive(SemTmp);

    bipLectOverrun = 0;
    }

  return(OK);
}

/*-----------------------------------------------*/
/*        fonction attachee au timer          */
/*-----------------------------------------------*/
```

```
static int bipLectSynchro()
{
  /* teste l'overrun */
  if (bipLectOverrun != 0)
    {
      bipLectOverrunFlag = 1;
    }
  /* autorise un nouveau calcul */
  semGive(SemTache);

  return(OK);
}


/*-----------------------------------------------------*/
/*    frequence en ms (1,2,3,4,5,6,7,8,9,10)   */
/*      generation de l'echantillonnage et        */
/*           sauvegarde des valeurs                */
/*-----------------------------------------------------*/
int  bipLecture(int axe)
{
  char name[64];
  int frequence = 10;         /* declaration et init de la variable frequence */
  int i,j;

  /* TEST SUR ARGUMENT */
  if ((axe < 0) || (axe > 15))
    {
      fprintf(stderr, "argument must be between 1 and 15 (and not %d)\n",
                axe);
      return(ERROR);
    }

  bipLectAxe = axe;

  /* gestion de l'init */
  if (bipLectInitFlag == ERROR)
    {
      bipHardInit();
      bipLectInitFlag = OK;
    }

  fprintf(stdout, "frequence d'echantillonnage = %d ms\n", frequence);

  /* CREATION */
  /* semaphore pour reperer les temps avec windview */
  SemTmp = semBCreate(SEM_Q_PRIORITY,SEM_FULL);

  /* semaphore pour activer la tache periodique */
  SemTache = semBCreate(SEM_Q_PRIORITY,SEM_EMPTY);

  /* tache periodique */
  if (taskSpawn("t_tache", 35, VX_FP_TASK|VX_STDIO,
                      10000, (FUNCPTR) bipPeriodTache,
                      0,0,0,0,0,0,0,0,0,0)==ERROR)
```

```
  {
   fprintf(stderr, "bipLectPeriod: Pb to launch bipPeriodTache\n");
   return(ERROR);
  }

 /* mise en place de l'interruption timer */
 sysAuxClkRateSet(1000 / frequence);   /* interruption toutes les 10 ms */
 sysAuxClkConnect(bipLectSynchro, 0);

 /* INITIALISATION */
 bipLectOverrun = 0;
 bipLectOverrunFlag = 0;
 bipLectExperimentation = 0;
 for (j=0;j<DIM_TABLEAU;j++)
  {
   for (i=0;i<BIP_DDL;i++)
        {
         bipLectArt[j][i] = 0.0;
         bipLectAdc[j][i] = 0.0;
        }
  }
 for (i=0;i<BIP_DDL;i++)
  {
   bipLectCodeur[i] = 0.0;
  }

 bipLectCpt = 0;

 /* Lance l'horloge */
 sysAuxClkEnable();

 printf("Debut d'attente\n");
 /* Duree de l'experimentation */
 taskDelay(50 * 60);   /* 60 = 1 sec */
 printf("Fin d'attente\n");

 /* declenche la fin de l'experimentation a la fin du cycle en cours */
 bipLectExperimentation = 1;

 taskDelay(60);      /* attente d'1 seconde (60=1s)  */

 /* Arrete l'horloge */
 sysAuxClkDisable();

 /* sauvegarde des resultats dans les fichiers */
 sprintf(name, "%s/potars%d.data", CHEMIN, axe);  /* attribue a name le chemin et
nom de fichier de sauvegarde */

 if (bipLectSaveFile(name, bipLectAdc) == ERROR)
  {
   printf("Probleme de generation de fichier\n"); /* message d'erreur */
  }

 sprintf(name, "%s/q%d.data", CHEMIN, axe);/* attribue a name le chemin et nom de
fichier de sauvegarde */
```

```
  if (bipLectSaveFile(name, bipLectArt) == ERROR)
    {
      printf("Probleme de generation de fichier\n");     /* message d'erreur */
    }

  /* printf("SemTmp = 0x%X - SemTache = 0x%X\n", (int)SemTmp, (int)SemTache);*/
  printf("Fin de l'experimentation\n\n");

  return(OK);
}

/*-------------------------------------------------*/
/*      fonction de sauvegarde des valeurs     */
/*-------------------------------------------------*/
static int bipLectSaveFile(char *fname, double tableau[DIM_TABLEAU][BIP_DDL])
{
  int i;
  FILE *fp;

  /* open the file */
  fp = fopen(fname,"w");
  if (!fp) return ERROR;

  /* enregistrement des valeurs */
  for (i=0; i<bipLectCpt-1;i++)
    {
      fprintf(fp, "%f\n", (float)tableau[i][bipLectAxe]);
    }

  /* fermeture du fichier */
  fclose(fp);

  return(OK);
}
```

ANNEXE 4 : TABLEAU RECAPITULATIF DES DUREES ET FREQUENCES DE L'EXPERIMENTATION :

n° articulation	*articulation correspondante*	*Situation*	Durée de la mesure	Echantillonnage
0	cheville axe frontal	Jambe Droite	30 s	10 ms
1	cheville axe sagital	Jambe Droite	30 s	10 ms
2	genou axe sagital	Jambe Droite	40 s	10 ms
3	hanche axe sagital	Jambe Droite	90 s	30 ms
4	cheville axe frontal	Jambe Gauche	30 s	10 ms
5	cheville axe sagital	Jambe Gauche	30 s	10 ms
6	genou axe sagital	Jambe Gauche	40 s	10 ms
7	hanche axe sagital	Jambe Gauche	90 s	30 ms
8	hanche axe vertical	Jambe Droite	90 s	30 ms
9	hanche axe frontal	Jambe Droite	90 s	30 ms
10	hanche axe vertical	Jambe Gauche	90 s	30 ms
11	hanche axe frontal	Jambe Gauche	90 s	30 ms
12	tronc axe vertical	Bassin	90 s	30 ms
13	tronc axe frontal	Lombaire	90 s	30 ms
14	tronc axe sagital	Lombaire	90 s	30 ms

ANNEXE 5 : EXEMPLE DE RESULTAT OBTENU AVEC MAPLE POUR UNE ARTICULATION

Genou droit 26/04/01

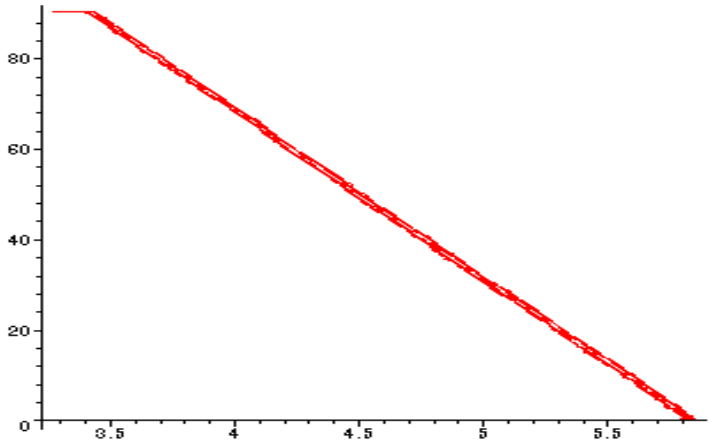
```
> # Exploitation des valeurs de potentiometres
> # Cree par : Pierre-Brice Wieber
> # Modifie par : Nathalie Cislo (11/04/2000)
> ##### Lecture de qd : tableau ligne de donnees articulaires;

>fd:=fopen('/local/projets/robotique/Bipede/utils/CalibPotars15/Mesures/q2.data',READ
):qd:=readdata(fd,float):fclose(fd):nops(qd);
```

3998

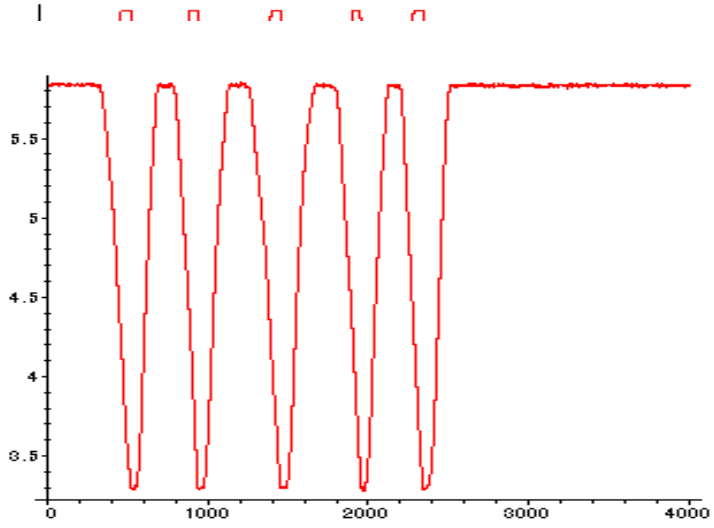
```
> ##### Lecture de po : tableau ligne de donnees potentiometres;
>fd:=fopen('/local/projets/robotique/Bipede/utils/CalibPotars15/Mesures/potars2.data',R
EAD):po:=readdata(fd,float):fclose(fd):N:=nops(po);
```

N :=3998

```
> ##### Transformation des donnees articulaires en degres
> qd:=[seq(qd[i]/3.14159*180,i=1..N)]:
> ##### Trace de qd en fonction de po
> plot([seq([po[i],qd[i]],i=1..N)]);
```




```
> plot([seq([i,qd[i]],i=1..N)]);
```

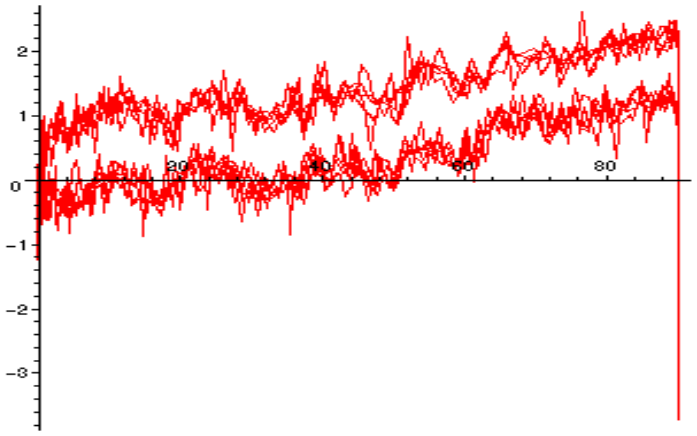


Trace d

```
> plot([seq([i, po[i]],i=1..N)]);
> ##### Recherche de la pente et de l'offset
> with(linalg):A:=leastsqrs(transpose([po,[1$N]]),qd);
```

A := [-36.66720360, 213.7709926]

```
> ##### Trace de l'erreur
> plot([seq([qd[i],qd[i]-A[1]*po[i]-A[2]],i=1..N)]);
```




```
> ##### Erreur maximale
> max(seq(abs(qd[i]-A[1]*po[i]-A[2]),i=1..N));
```

3.7588112

```
> ##### Amplitude maximale
> max(seq(qd[i],i=1..N))-min(seq(qd[i],i=1..N));
```

90.10382002

```
> ##### Min des potars
> min(seq(po[i],i=1..N));
```

3.273010

```
> ##### Max des potars
> max(seq(po[i],i=1..N));
```

5.843201

ANNEXE 6 : PROGRAMME DONNANT LA POSITION ANGULAIRE A L'AIDE DE POTENTIOMETRES

```
                /*-------------------------------------------------------------*/
                /*  Position absolue instantanee de chaque articulation*/
                /*-------------------------------------------------------------*/
#include <stdio.h>
#include <math.h>

#ifdef VXWORKS
#include <taskLib.h>
#include <tickLib.h>
#include <sysLib.h>
#include <semLib.h>
#endif

#include "hardBip.h"
#include "utilBip15.h"

#include "BipPotars.h" /*Fichier contenant les valeurs des offsets et des pentes*/

/*---------------------------------------------*/
/*          Definition des variables           */
/*---------------------------------------------*/
#define BIP_DDL 15

/* POTAR : tableau de passage des entrees module_ip aux entrees bipede */
static int bipLectPotarCor[BIP_DDL] = {3, 11, 12, 4, 1, 9, 10, 2, 7, 6,
                                       5, 13, 14, 8, 16};
static double bipLectAdc[BIP_DDL];  /*tableau des valeurs des potars */
static double bipPosAbs[BIP_DDL];   /*tableau des valeurs absolues des articulations */
static int bipTestPosAbsInitFlag = ERROR;   /* variable gerant l'initialisation */

/*---------------------------------------------*/
/*         fonction de travail                 */
/*---------------------------------------------*/
void bipTestPositionAbsolue()
{
  int bipLectAxe;
  float ConvRad_Deg=180/3.14159;
/* tableau des coeff directeurs */
  double Pot[BIP_DDL]={ PotCheD_F, PotCheD_S, PotGenD_S, PotHanD_S,
                        PotCheG_F, PotCheG_S,     PotGenG_S, PotHanG_S,
                        PotHanD_F, PotHanD_V, PotHanG_F, PotHanG_V,
                        PotLomb_V, PotLomb_F, PotLomb_S};
/* tableau des offsets */
  double Off[BIP_DDL]={ OffCheD_F, OffCheD_S, OffGenD_S, OffHanD_S,
                        OffCheG_F, OffCheG_S, OffGenG_S, OffHanG_S,
                        OffHanD_F, OffHanD_V, OffHanG_F, OffHanG_V,
                        OffLomb_V, OffLomb_F, OffLomb_S};
  /* gestion de l'init */
  if (bipTestPosAbsInitFlag == ERROR)
    {
      bipHardInit();
```

```
    bipTestPosAbsInitFlag = OK;
   }
  /* lecture des potars */
  for(bipLectAxe = 0; bipLectAxe < 15; bipLectAxe++)
   {
    if (bipHardGetAnalog(bipLectPotarCor[bipLectAxe]+15,
                         &bipLectAdc[bipLectAxe]) != OK)
        {
         fprintf(stderr, "bipHardGetAnalog retourne ERROR\n");
        }
    else
        {
         printf("valeur potar  : %f \t potar associe :  %d \n",
                 bipLectAdc[bipLectAxe], bipLectAxe);
        }
   }
   /* calcul et affichage des positions absolues de chaque articulation*/
  for(bipLectAxe = 0; bipLectAxe < 15; bipLectAxe++)
   {
    bipPosAbs[bipLectAxe]=(bipLectAdc[bipLectAxe] * Pot[bipLectAxe]
                          + Off[bipLectAxe])*ConvRad_Deg;
    printf("Position angulaire de l'articulation %d\t : %f degres\n",
            bipLectAxe, bipPosAbs[bipLectAxe]);
   }
  printf("\n");
}
```

SOMMAIRE

Introduction

Le premier tome a été consacré à la description de l'entreprise, ainsi qu'à la présentation du sujet de stage et de l'objectif à atteindre. Nous avions également fait un premier compte rendu des choix technologiques adoptés et des résultats obtenus à mi-stage.

Pour resituer l'état d'avancement, nous allons faire un rapide rappel des résultats obtenus jusqu'ici. Le choix technologique retenu pour connaître la position du robot à la mise sous tension a été l'utilisation de potentiomètres. En théorie, ceux-ci délivrent une tension proportionnelle à la position angulaire de chaque articulation. La première étape a donc consisté à réaliser une calibration de tous ces potentiomètres. Ensuite, nous avons fait une estimation des erreurs engendrées, pour valider ou non leur utilisation. En effectuant une valeur moyennée sur 50 mesures, nous obtenons une erreur maximale de ±0,2°, ce qui est inférieur à la précision souhaitée (qui est de ¼ de degrés). Il reste tout de même à estimer la robustesse et la répétabilité.

Ce second tome a pour objectif d'exposer le travail réalisé en seconde partie de stage. Dans un premier temps, nous ferons une présentation du contrôleur implémenté sur le robot. Ce contrôleur permet le suivi de trajectoire. Nous présenterons ensuite les manipulations et expérimentations effectuées et nous ferons un exposé clair des résultats obtenus. En dernier point, nous aborderons les problèmes rencontrés lors du projet et nous exposerons les perspectives envisagées.

1 – Limitation des mouvements

1-1 INTRODUCTION :

D'un point de vue mécanique, les mouvements des articulations sont limités. Afin d'éviter un choc en butée mécanique, une sécurité électrique a été rajoutée pour arrêter le mouvement. Si le robot évolue sur le sol et que les butées électriques sont atteintes, le circuit puissance est coupé et le robot s'effondre. Pour éviter cette détérioration du robot, une sécurité logicielle a été ajoutée (cette sécurité évite le déclenchement des sécurités électriques).

1-2 DIFFERENTES BUTEES :

➪ Les butées mécaniques correspondent au débattement de l'articulation entre deux contacts mécaniques sur une des pièces du robot.

➪ Les butées électriques correspondent au débattement autorisé avant que l'articulation ne déclenche un capteur fin de course. Lorsque ce capteur se déclenche, le moteur est coupé.

➪ Du fait que les moteurs soient coupés lorsque le robot arrive en butée électrique, il est nécessaire de placer des butées logicielles qui garantissent que le robot n'atteigne pas ces butées électriques.

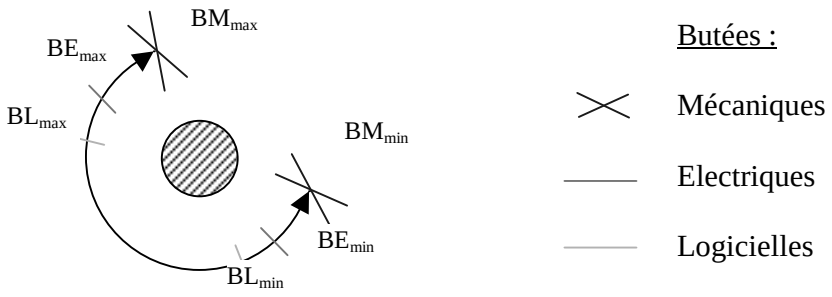


Schéma du positionnement des butées sur le débattement d'une articulation

1-3 MISE A JOUR DES VALEURS :

En utilisant les programmes précédemment réalisés (programme en annexe 6 Tome 1), nous avons remis à jour les valeurs des butées mécaniques. Nous avons positionné les articulations en butée mécanique et nous avons obtenu les valeurs angulaires en utilisant les potentiomètres.

Nous avons également remis à jour les butées électriques en réalisant un câblage adapté et un programme qui affiche les valeurs articulaires dès qu'une butée électrique est atteinte. (Cf. annexe 1 pour le câblage et le programme). Du fait que ces valeurs soient obtenues par les potentiomètres, elles sont estimées à 0.2 degrés près.

Pour les mouvements combinés, plusieurs configurations (de positionnement) sont possibles, il est donc impossible de déterminer une valeur unique. Dans un premier temps, nous avons décidé d'effectuer soit un mouvement purement frontal, soit purement sagittal. Pour les chevilles, il est impossible d'atteindre les butées électriques, si le mouvement est purement frontal ou sagittal. Les valeurs fournies sont donc juste à caractère informatif, il s'agit d'une configuration quelconque (c'est-à-dire actions frontale et sagittale combinées) où la butée électrique est atteinte. Les valeurs frontale et sagittale sont à considérer ensemble.

Une fois les valeurs électriques obtenues, nous avons pu réactualiser les valeurs software. Ces valeurs sont décalées d'environs 2 degrés par rapport aux valeurs électriques.

2 – Présentation du contrôleur

2-1 BUT DU CONTROLEUR :

Un contrôleur a été réalisé pour permettre au robot de suivre des trajectoires articulaires. Ainsi, le robot pourra évoluer sur le sol en suivant des trajectoires prédéfinis par un simulateur.

Ce contrôleur a été implémenté à l'aide du logiciel ORCCAD (environnement de programmation), et dans un premier temps pour un robot en simple support droit uniquement.

Le contrôleur peut être décomposé en 3 parties :

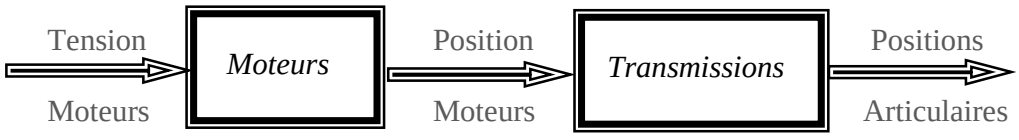
- communication avec le système,
- procédure d'initialisation,
- poursuite de trajectoire.

2-2 COMMUNICATION AVEC LE SYTEME :

Le premier rôle du contrôleur est d'envoyer des signaux de commande pour agir sur le robot. Il est également informé de l'état du robot par l'intermédiaire de capteurs.

d) *Communication avec les actionneurs du robot :*

Les actionneurs sont constitués de moteurs CC et de réducteurs de type Harmonic Drive ou transmissions spéciales. L'asservissement en position des articulations se fait par une commande en tension des moteurs. Pour connaître la relation entre tensions moteurs et positions articulaires, il est donc nécessaire de connaître le modèle de la chaîne de transmission.



Rapport de réduction

A partir de la position relative en top codeurs et de la position de départ, une fonction appelée *bipGetEncoders* calcule la position absolue (en radians) du moteur. La position de départ est obtenue par la valeur du point milieu des potentiomètres.

Une fois ces valeurs obtenues pour chaque moteur, il s'agit de les retranscrire en positions articulaires. Pour obtenir ceci, nous utilisons une fonction appelée

CalcPosRatio, qui effectue une interpolation d'un tableau de correspondance (pour les articulations de type vis à rouleau) entre la position moteur et la position articulaire. Pour les transmissions parallèles (mouvement utilisant l'association de 2 moteurs), c'est-à-dire les chevilles et le tronc, la position articulaire est reliée à la somme ou à la différence des positions moteurs.

e) *Communication avec les capteurs du robot :*

Il est également possible de lire les informations sur les différents capteurs disponibles tels que les capteurs de forces, inclinomètre, ou capteur ultrasons.

2-3 PROCEDURE D'INITIALISATION :

L'objectif de l'initialisation est de connaître la position dans laquelle se trouve le robot au démarrage. Deux solutions ont été mises en œuvre, soit le positionnement du robot en position connue (Initialisation manuelle), soit l'utilisation des potentiomètres (Initialisation automatique).

➪ **Initialisation manuelle :**

Pour réaliser cette initialisation, il s'agit de positionner le robot dans une position connue. Jusqu'à présent seule la position zéro est considérée comme parfaitement connue.

➪ **Initialisation automatique :**

Cette initialisation consiste à estimer la position du robot au démarrage. Comme, nous l'avons présenté dans le tome 1, la solution retenue est l'utilisation des potentiomètres qui fournissent une tension proportionnelle à la position angulaire de chaque articulation.

2-4 PROCEDURE DE POURSUITE DE TRAJECTOIRES :

a) *Différentes phases de support :*

D'un point de vue mécanique, le robot en phase de locomotion est un système mécanique de corps rigides soumis à des contraintes unilatérales. Ces contraintes sont des inégalités qui traduisent la non-interpénétrabilité des corps rigides en présence de frottements secs. On peut isoler quatre configurations principales pour le robot, en fonction du nombre de pieds en contact avec le sol et selon le pied en contact avec le sol.

➪ **Robot suspendu au-dessus du sol :**

Dans les phases de réglage et d'ajustement des applications, le robot n'est pas posé sur le sol. De même, jusqu'ici l'initialisation du robot, c'est-à-dire l'étape pendant laquelle on évalue la position absolue du robot, se fait robot suspendu. Dans ces deux cas, la caisse qui sert de tronc est soutenue par le haut par des sangles accrochées à un palan mécanique que l'on peut actionner par une télécommande pour descendre ou monter le robot.

➪ **Robot en mouvement sur le sol :**

Lorsque le bipède se déplace au sol, il peut avoir un seul de ses pieds au sol (simple support) ou les deux en même temps (double support).
On a donc trois configurations différentes possibles :

- simple support pied droit,
- simple support pied gauche,
- double support pied droit ou pied gauche en avant.

b) Détection de support :

Lorsque le robot évolue sur le sol, la détection de support joue un rôle très important. Elle permet de savoir dans quelle phase de support se trouve le robot et de détecter les éventuels impacts avec le sol ou un obstacle. De plus, on peut également observer les déplacements du centre de pression et vérifier que l'équilibre est conservé.

La détection de support s'appuie sur les informations fournies par les capteurs d'effort. Les phases de support considérées sont : le double support, le simple support droit, le simple support gauche, les transitions. Les transitions correspondent aux cas où la somme des forces de pression est inférieure au poids total du robot, c'est-à-dire lorsque le robot n'est que partiellement posé sur sol (en cours de dépose ou en cours de levage).

Sur chaque pied, des capteurs sont situés au talon, à l'avant droit et à l'avant gauche, ils sont numérotés de 1 à 6. Chaque capteur délivre une valeur qui correspond à la force exercée par le tibia sur le capteur. Ainsi il est possible de déterminer dans quelle phase se trouve le robot en utilisant ces valeurs des capteurs. Il est également possible de connaître la position du centre de pression (point en lequel le moment des efforts de pression est nul). Lorsque ce point est situé sous la semelle du pied, le robot est considéré en équilibre.

c) Contrôle et commande :

➪ **Méthode :**

Le but est de permettre au robot la poursuite des trajectoires articulaires qd. Ces trajectoires pourront être calculées a priori ou en-ligne. Pour cela, nous utilisons une loi de commande de type proportionnel-dérivé associé à une compensation de gravité et une compensation de frottement. On calcule dans un premier temps la partie PD+Gravité pour déterminer les couples articulaires Γ:

$$\Gamma = K_p * (q_d - q) + K_v * (\dot{q}_d - \dot{q}) + \hat{G}(q)$$

avec K_p et K_v des vecteurs de constantes positives, *qd* la trajectoire désirée que l'on poursuit et *G* une estimation du vecteur gravité.

Les moteurs que nous utilisons sont commandés en tension *U*. Nous avons transformé Γ en vecteur de tensions *U'* grâce aux modèles des moteurs et nous avons introduit une compensation des frottements :

$$U = U' + F * sign(\dot{\theta}_m)$$

où F est le vecteur des constantes de frottements et $\dot{\theta}_m$ le vecteur des vitesses moteurs.

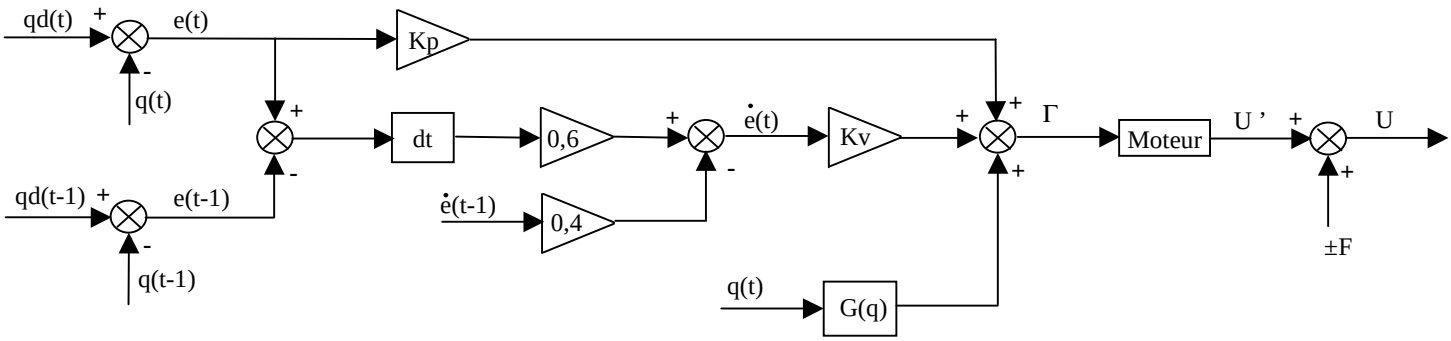
➪ **Evaluation des frottements secs :**

Pour estimer les frottements secs de chaque moteur, nous avons réalisé un programme qui injecte une consigne s'incrémentant à une période choisie avec un pas souhaité (ici 0,3 volts) jusqu'à ce que le moteur tourne d'un nombre de pas désirés (ici 200 pas codeurs).

Ce programme teste chaque moteur l'un après l'autre. On obtient finalement la valeur de la tension à injecter à chaque moteur pour que ceux-ci compensent les frottements secs.

Il est nécessaire de tester les frottements secs pour les 2 sens de rotation des moteurs. Il est vrai que la position de départ du robot influence sur la valeur de ces frottements. Le fait que les moteurs aient été rodés modifie également ces valeurs. Nous avons donc choisi une position de départ pour laquelle la gravité ne joue pas un rôle trop important, c'est-à-dire une position verticale du robot. Le programme ainsi que les résultats obtenus sont exposés en annexe 2.

➪ **Schéma d'asservissement :**



Période d'échantillonnage de 9 ms.

- *Vecteur gravité :*

Le vecteur gravité G(q) dépend de la phase de support considérée. Ce vecteur est donc estimé suivant que le robot soit en simple support droit, en robot suspendu ou en simple support gauche. Ce vecteur a été déterminé de manière formelle par simulation et ensuite validé par expérimentation.

Il se peut que le robot se retrouve en double support, mais cette phase est concidérée comme courte. La méthodologie utilisée est donc un transfert linéaire du poids d'un pied sur l'autre. Nous supposons dans le cas où l'on est en train de soulever ou de poser le robot que le poids arrive de manière linéaire sur le pied au sol. Cela revient à dire que l'on passe linéairement d'un modèle à l'autre. On écrit donc que :

$$\hat{G} = Lambda0 * G0 + Lambda1 * G1$$

où Lambda0 correspond au rapport entre le poids supporté par le pied droit et le poids total du robot. Et Lambda1 = 1-Lambda0.

- *Valeurs des gains :*

En ce qui concerne les valeurs des *Kp*, on dispose de 2 jeux de gains suivant que le robot soit suspendu ou au sol. Dans le cas de la transition, on utilise une combinaison linéaire des 2 jeux de gain en fonction du poids qui se déplace sur le pied de support.

- *Calcul de l'erreur en vitesse :*

Cette erreur est déterminée en fonction de l'erreur de position *e(t)* par un filtre :

$$\dot{e}(t) = 0.4 * \dot{e}(t-1) + 0.6 * \frac{(e(t) - e(t-1))}{dt}$$

Ainsi, tous les éléments sont calculés pour pouvoir déterminer le couple de commande relatif au PD+compensation de gravité. Il s'agit ensuite de convertir ces couples de commande en tensions moteurs à partir des rapports de réduction et des constantes moteurs fournies par les constructeurs.

On rappelle les relations entre positions moteurs θ et positions articulaires q, ainsi que la relation entre couples moteurs C_m et articulaires C_a :

$$J * \dot{\theta} = \dot{q}$$
$$C_m^T * \dot{\theta} = C_a^T * J * \dot{\theta}$$

D'où :

$$C_m = J^T * C_a$$

où J est la matrice des rapports de réduction. Dans le cas des transmissions de type parallèle :

$$J = \begin{pmatrix} \text{ratio}_{\text{frontal-exterieur}} & \text{ratio}_{\text{frontal-interieur}} \\ \text{ratio}_{\text{sagittal-exterieur}} & \text{ratio}_{\text{sagittal-interieur}} \end{pmatrix}$$

On ajoute alors la compensation de frottements, on obtient la tension de commande des moteurs. La compensation utilise le signe de la vitesse moteur désirée qui est calculée à partir des positions articulaires désirées à l'aide d'un filtre identique à celui utilisé pour le calcul de l'erreur en vitesse.

2-5 ECRITURE D'UNE TRAJECTOIRE :

a) *Méthode :*

On utilise deux formats de fichiers de trajectoires. Le premier format contient des points de passage entre lesquels on interpole pour trouver les positions à chaque instant. Le second contient les positions articulaires désirées à chaque instant.

b) *Trajectoire polynomiale :*

Les fichiers utilisés par le module contiennent des points de passage souhaités pour les différentes articulations. On calcule ensuite un polynôme de degré 5 reliant les points de passage et assurant la vitesse désirée à chaque point de passage. On peut ainsi définir des mouvements de façon très simple et rapide pour le robot lorsqu'il est suspendu (réglages de gains, rodage...). Le format des fichiers est :

```
# Trajectoire
LOOP X Y Z
TIME t
POSR a b c d
VELR 0 0 0 0
POSL e f g h
VELL 0 0 0 0
POSB i j k l m
VELB 0 0 0 0 0
POST n o
VELT 0 0
```

Les trois valeurs qui suivent le mot clef *LOOP* sont :
X : le nombre de répétitions de la trajectoire,
Y : le point de la trajectoire sur lequel on recommence les cycles,
Z : la durée de ralliement du premier point d'un cycle à partir du dernier point du cycle précédent.

Le mot clef *TIME* est suivi de la durée de ralliement d'un point à partir du point précédent.

Les quatre valeurs qui suivent le mot clef *POSR* sont :
a : position articulaire de la cheville droite dans l'axe frontal,
b : position articulaire de la cheville droite dans l'axe sagittal,
c : position articulaire du genou droit,
d : position articulaire de la hanche droite dans l'axe sagittal.

Les quatre valeurs qui suivent le mot clef *POSL* sont :
e : position articulaire de la cheville gauche dans l'axe frontal,
f : position articulaire de la cheville gauche dans l'axe sagittal,
g : position articulaire du genou gauche,
h : position articulaire de la hanche gauche dans l'axe sagittal.

Les cinq valeurs qui suivent le mot clef *POSB* sont :
i : position articulaire de la hanche droite dans l'axe frontal,
j : position articulaire de la hanche droite dans l'axe vertical,
k : position articulaire de la hanche gauche dans l'axe frontal,
l : position articulaire de la hanche gauche dans l'axe vertical,
m : position articulaire du tronc dans l'axe vertical.

Les deux valeurs qui suivent le mot clef *POST* sont :
n : position articulaire du tronc dans l'axe frontal,
o : position articulaire du tronc dans l'axe sagittal.

Toutes ces différentes valeurs ne doivent pas dépasser les limites articulaires logicielles.

Les mots clefs VELR, VELL, VELB, VELT sont suivis des valeurs désirées de vitesses pour les différentes positions désirées. Cette fonctionnalité n'ayant pas été testée, les valeurs des vitesses seront toutes nulles.

c) ***Trajectoire V0:***

Le principe consiste à décrire des postures à l'aide de fonctions de sortie, à interpoler entre ces postures pour obtenir un mouvement puis à convertir ces trajectoires en trajectoires articulaires. Les mouvements sont générés de manière à assurer l'équilibre statique du robot au sol. Cette méthode permet de générer l'ensemble des positions articulaires à chaque instant pour le robot.

```
# Format
V0
# NbArt
15
# NbPts
20
# Cycle
300
# Rebouclage
1
# Temps
12
0.1 0 0 0.1 0 0 0.3 0 0 0.1 0 0 0.0 0 0 0.0 0 0 0.6 0 0 0.6 0 0 0.1 0
0 -0.1 0 0 0.8 0 0 0 0 0 0 0 0 0.8 0 0 0.1 0 0.1
0.1 0 0 0.1 0 0 0.3 0 0 0.1 0 0 0.0 0 0 0.0 0 0 0.6 0 0 0.6 0 0 0.1 0
0 -0.1 0 0 0.8 0 0 0 0 0 0 0 0 0.8 0 0 0.1 0 0.1
0.1 0 0 0.1 0 0 0.3 0 0 0.1 0 0 0.0 0 0 0.0 0 0 0.6 0 0 0.6 0 0 0.1 0
0 -0.1 0 0 0.8 0 0 0 0 0 0 0 0 0.8 0 0 0.1 0 0.1
```

On a une liste de valeurs rangées dans l'ordre de numérotation des articulations et regroupées par 3 : position articulaire, vitesse articulaire, accélération articulaire désirées. Vitesse et accélération ne sont pas utilisées dans la version actuellement implémentée.

3 – Présentation de l'étude

3-1 CONTEXTE :

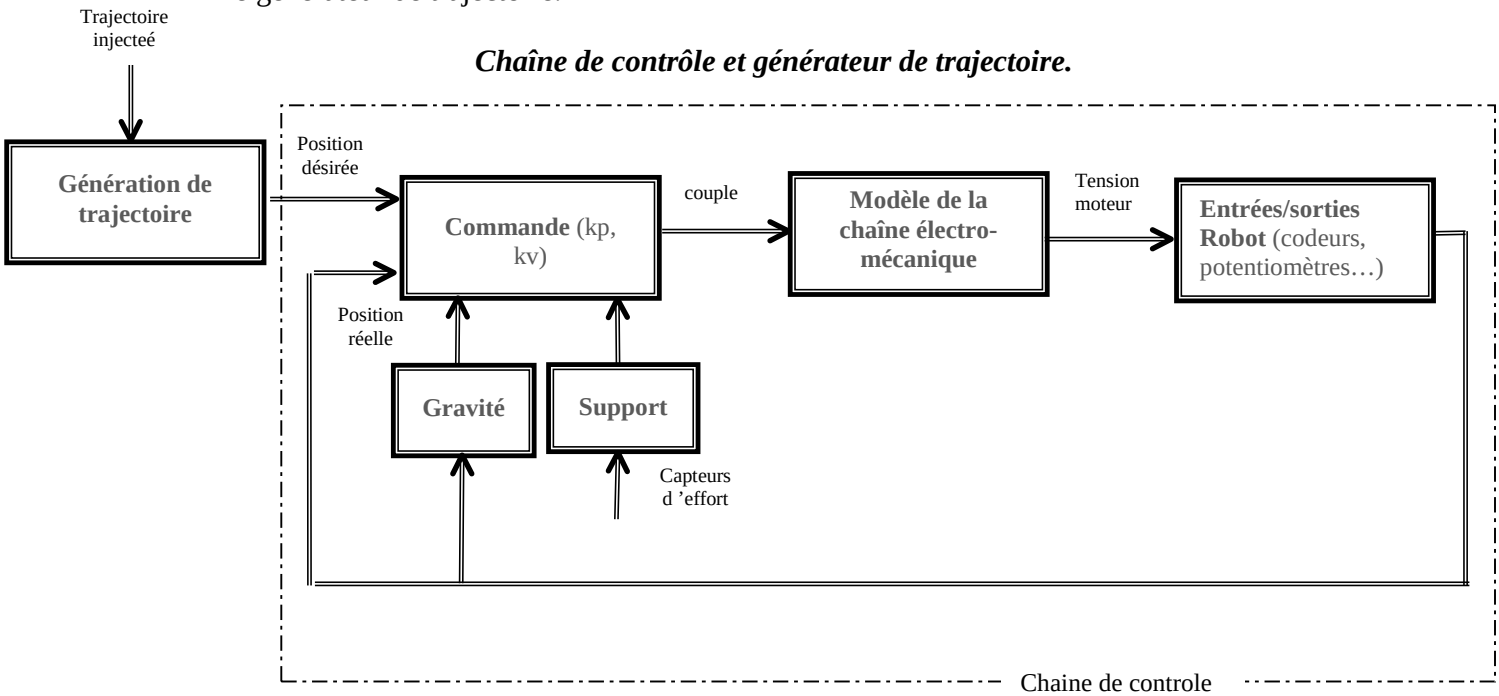
Dans le cadre du projet BIP, deux prototypes de robots bipèdes ont été conçus et réalisés. Le robot exposé à Hanovre est actuellement au LMS de Poitiers, et le second robot est utilisé à l'INRIA. L'état d'avancement des deux robots n'a pas évolué de la même manière. En effet, pour l'exposition universelle, le robot exposé pouvait déjà se tenir en équilibre sur un pied. Toute la partie contrôleur avait été testée et validée, cependant les tests effectués avaient toutefois été réalisés dans l'urgence. Aucune méthodologie n'avait donc été rédigée.

Pour le robot de l'INRIA, le contrôleur n'avait pas encore été testé à mon arrivée. Il a donc fallu élaborer une méthodologie de tests (répétable) qui garantisse que le contrôleur implémenté est exploitable.

3-2 VALIDATION DU CONTROLEUR :

La démarche suivie consiste à vérifier progressivement la chaîne de contrôle et le générateur de trajectoire.

Chaîne de contrôle et générateur de trajectoire.



Nous désirons tester de manière fonctionnelle chaque module. On appelle module chaque boite du schéma ci-dessus (exemple : Gravité, Support …)

Le logiciel Orccad permet la sauvegarde à la période d'échantillonnage (ici 9 ms) de la position réelle, de la position désirée, du couple, de la tension moteur, et des valeurs des capteurs d'effort. En analysant ces données, nous allons pouvoir valider ou non chaque module.

Procédure de tests :

Par mesure de sécurité, les étapes 1 à 5 suivantes sont réalisées avec des paramètres diminués par rapport à ceux du robot d'Hanovre. Les tests 1 à 3 ont été réalisés avec le bouton d'arrêt d'urgence enfoncé, car aucun mouvement du robot n'est nécessaire. Jusqu'au test 6, nous avons utilisé l'init manuelle, l'init automatique sera testée ultérieurement (test 7).

1. Le module « Support » renvoie un entier suivant la phase de support du robot. (0 : robot en l'air, 1 : simple support droit, 2 : simple support gauche, 3 : double support). Toutes les expérimentations faites pour la validation du contrôleur ont été réalisées robot en l'air, donc nous avons vérifier à chaque fois que le module renvoyait bien l'état 0. Ce module sera réellement validé lorsque nous effectuerons de phase de pose au sol.

2. Tester le bon comportement de la loi de commande dans le cas particulier d'une trajectoire nulle.

3. Test du générateur de trajectoire.

4. Valider la compensation de gravité.

5. Rodage.

6. Réglage des paramètres.

7. Test de l'init automatique.

8. Validation globale robot en l'air.

Les détails de ces différents tests sont exposés dans le chapitre suivant.

3-3 TESTS D'EQUILIBRE DU ROBOT :

Une fois tous les modules (Hardware et Software) validés et les paramètres réglés, nous avons rejoué les postures réalisées sur le robot d'Hanovre et également testé des postures sur support pied gauche qui jusqu'ici n'avaient jamais été implémentées. Les détails des expérimentations sont exposés au chapitre 5.

4 – Validation du contrôleur

4-1 INTRODUCTION :

Dans un premier temps, nous avons joué des trajectoires avec le robot en l'air et avec l'arrêt d'urgence enfoncé. Il s'agit donc d'écrire et de jouer des trajectoires de type polynomial qui valident le bon fonctionnement du contrôleur. Ensuite nous avons effectué une phase de réglage. Puis pour finir, nous avons validé le contrôleur dans son ensemble.

4-2 TEST DE LA CHAINE DE CONTROLE :

But et procédé :

Il s'agit de tester le bon comportement de la loi de commande dans le cas particulier d'une trajectoire nulle.

Le procédé consiste à n'avoir aucune erreur à corriger et aucune gravité, pour que l'on puisse valider que le couple et la tension envoyée au moteur soient faibles pour ces conditions. A la mise sous tension, le robot est considéré en position zéro. Il s'agit de lui injecter une trajectoire nulle, pour qu'il n'y ait aucune erreur à corriger. Pour avoir une gravité quasiment nulle, nous avons positionné le robot en zéro mécanique (seule position où la gravité n'intervient pratiquement pas).

Ce test peut être réalisé avec le bouton d'arrêt d'urgence enfoncé, car la position réelle du robot est toujours considérée comme nulle. Nous souhaitons juste valider que les tensions envoyées aux moteurs soient faibles (voire nulle si le vecteur gravité n'intervient pas). Vu que l'erreur et la trajectoire restent nulles, la tension envoyée doit rester constante.

Il subsiste le problème des genoux, qui en position zéro, sont considérés comme en butée logicielle. Il est donc nécessaire de les retirer de ces butées, pour pouvoir lancer la trajectoire.

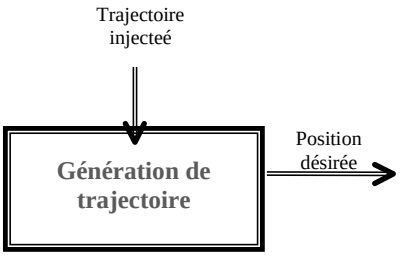
Résultats :

Nous obtenons bien des tensions très faibles pour les articulations, sauf pour les lombaires où l'on atteint 0,5 Volts (ceci s'explique par le module de gravité qui agit forcément sur certaines articulations). Pour les genoux, nous avons une tension plus conséquente, ce qui s'explique par le fait que l'erreur ne peut être nulle.

4-3 TEST DU GENERATEUR :

But et procédé :

Dans le but de tester le générateur de trajectoire, nous avons réalisé une trajectoire qui relie 2 positions totalement différentes. On s'intéresse au générateur de trajectoire, donc seulement à l'entrée et à la sortie de ce module. Il s'agit de comparer les valeurs des positions désirées aux valeurs données dans la trajectoire injectée. Les positions réelles du robot ne sont pas nécessaires, seul le module générateur de trajectoire nous intéresse. Nous pouvons donc jouer cette trajectoire avec l'arrêt d'urgence enfoncé.



Pour ce test, il s'agit de sauvegarder les positions désirées de chaque articulation. Comme nous l'avons présenté précédemment, le module utilise des fichiers qui contiennent des points de passage souhaités pour les différentes articulations. Un polynôme de degré 5 reliant les points de passage est ensuite calculé.

Résultats :

Pour chaque articulation, nous avons donc analysé les positions désirées que fournit le générateur de trajectoire, et en effet ces valeurs correspondent bien avec les valeurs données dans la trajectoire. Nous pouvons donc considérer que ce module fonctionne parfaitement.

4-4 ETUDE DE LA GRAVITE :

But et procédé :

Le but de ce test est de valider la compensation de gravité. Ce module a été établi de manière formelle et par ce test nous souhaitons le valider de manière expérimentale.

Les conditions du test consistent à régler les Kp à 0 et à injecter une trajectoire nulle. Ainsi l'erreur entre la position désirée et la position réelle du robot n'intervient pas dans la commande (elle est nulle quelque soit la position du robot). Seule la gravité joue un rôle.

Ce test est à réaliser avec le bouton d'arrêt d'urgence désactivé car la partie puissance entre en compte pour maintenir le robot. On souhaite observer si le robot compense la gravité, c'est-à-dire s'il se maintient dans n'importe quelle position. Une fois la trajectoire lancée, on bouge manuellement une articulation et on la relâche pour observer si la gravité est compensée.

Résultats :

Ce test est à réaliser pour chaque articulation du robot. Les résultats observés ont été très concluants, en effet quelque soit sa position, le robot maintient sa posture. Le module gravité a donc été validé de manière expérimentale.

4-5 RODAGE DES JAMBES ET DU TRONC :

***But et procédé* :**

Nous avons effectué une étape de rodage des moteurs. La structure du robot de l'INRIA est constituée pour la partie du bas de la structure de la version du robot à 8 degrés de libertés et pour la partie haute d'une structure nouvelle. Nous avons donc rodé en particulier les moteurs de la partie haute, les moteurs du bas ayant déjà été utilisés.

Le rodage consiste à injecter une trajectoire avec un débattement relativement ample à toutes les articulations du robot sans toutefois trop se rapprocher des butées. Les Kp sont réglés de moitié par rapport à ceux réglés après Hanovre.

Le mouvement injecté est le suivant :
- mouvement sagittal de la cheville,
- retour au point zéro,
- mouvement frontal de la cheville,
- retour au point zéro,
- mouvement de la hanche et du genou en simultané,
- retour au point zéro,
- mouvement frontal des hanches,
- retour au point zéro,
- mouvement vertical des hanches et vertical du tronc en simultané,
- retour au point zéro,
- mouvement vertical des lombaires,
- retour au point zéro,
- mouvement sagittal du tronc,
- retour au point zéro.

Résultats :

Dans un premier temps, le mouvement du tronc dans le plan sagittal ne fonctionnait pas, le moteur arrière droit (13) ne fonctionnant pas. La fonction *bipPutCurrent* (fonction qui envoie le courant) comportait un mauvais argument correspondant à ce moteur.

Une fois ce problème résolu, nous avons rejoué la trajectoire. Elle semble correcte pour un rodage mais cependant le réglage des Kp n'est pas suffisamment affiné. Il subsiste tout de même un léger problème sur la hanche gauche dans le mouvement sagittal. On a un léger décalage de 0.02 rad soit 1,2 degrés par rapport à la hanche droite, ce qui n'est pas négligeable. (environs 2 cm d'écart entre les pointes des pieds). Ce problème sera réglé lors de la phase de réglage des kp.

4-6 REGLAGE DES PARAMETRES :

But et procédé :

Le but est de régler les PD+Gravité de chaque articulation en garantissant une erreur entre la position désirée et la position réelle la plus faible possible et en assurant que le robot ne vibre pas. Cette étape est à réaliser avec le robot en l'air et avec des valeurs de frottements secs déjà évaluées.

Pour chaque articulation, il s'agit d'injecter successivement une trajectoire du type :
- trajectoire courte pendant un temps long,
- trajectoire courte pendant un temps court,
- trajectoire longue pendant un temps long,
- trajectoire longue pendant un temps court.

On injecte cette trajectoire avec différentes valeurs des *Kp*.

Traces à enregistrer :

- les couples,
- les positions désirées,
- les positions réelles.

Procédure de sauvegarde des valeurs, de visualisation et d'analyse :

Les valeurs sont stockées sous un répertoire sous forme d'un tableau en fonction du temps. Il s'agit ensuite de les filtrer et de les exporter. Pour cela, un fichier appelé datfilter permet d'exploiter des données au cours d'une manipulation. Un script shell est appelé par la commande datfilter, il permet de diviser les colonnes du fichier de données en autant de fichiers unidimensionnels pour permettre leur exploitation sous Maple.

Ensuite, en utilisant Maple, nous avons affiché les valeurs sous forme de courbes et effectuer des calculs tel que l'erreur maximum entre la position désirée et la position réelle. (le fichier maple et un exemple de résultat obtenu sont donnés en annexe 3)

Résultats :

Pour chaque articulation, nous avons progressivement augmenté la valeur du Kp pour atteindre une valeur qui semble la plus appropriée pour à la fois avoir une erreur de position faible et ne pas avoir de vibrations.

Nous avons retrouvé des valeurs avoisinant celles implémentées sur le robot d'Hanovre. Pour le réglage du second jeu de gain, c'est-à-dire lorsque le robot est en phase de simple support, nous avons diminué légèrement les valeurs des kp du robot d'Hanovre, et nous les modifierons si nécessaire lors des essais d'équilibre suivant la réaction du robot.

4-7 TEST DE L'INIT AUTOMATIQUE :

But et procédé :

Ce test consiste à valider l'utilisation ou non de l'init automatique. Le procédé est simple, on positionne le robot dans une posture quelconque, puis on lui injecte une trajectoire qui le place en position zéro (sauf pour les genoux). La position zéro est la seule position parfaitement connue. Ce test nécessite bien sur le bon réglage des Kp. La valeur angulaire de chaque articulation est affichée avant le lancement de la trajectoire, on peut donc dans un premier temps évaluer visuellement si cela correspond avec la position du robot.

Résultats :

Les erreurs de positionnement obtenues par rapport à la position zéro coïncident avec les erreurs que nous avions à la fin du paramétrage. L'init automatique est donc utilisable.
Par la suite, l'init automatique sera utilisée à chaque expérimentation, ceci simplifie considérablement la tache de l'opérateur.

4-8 TEST GLOBAL :

But et procédé :

Ce test permet la validation du contrôleur dans son ensemble. Le principe est simple, il s'agit d'injecter une trajectoire quelconque au robot (par exemple la trajectoire du rodage), et d'analyser les erreurs obtenues entre la position désirée et la position réelle du robot.

Résultats :

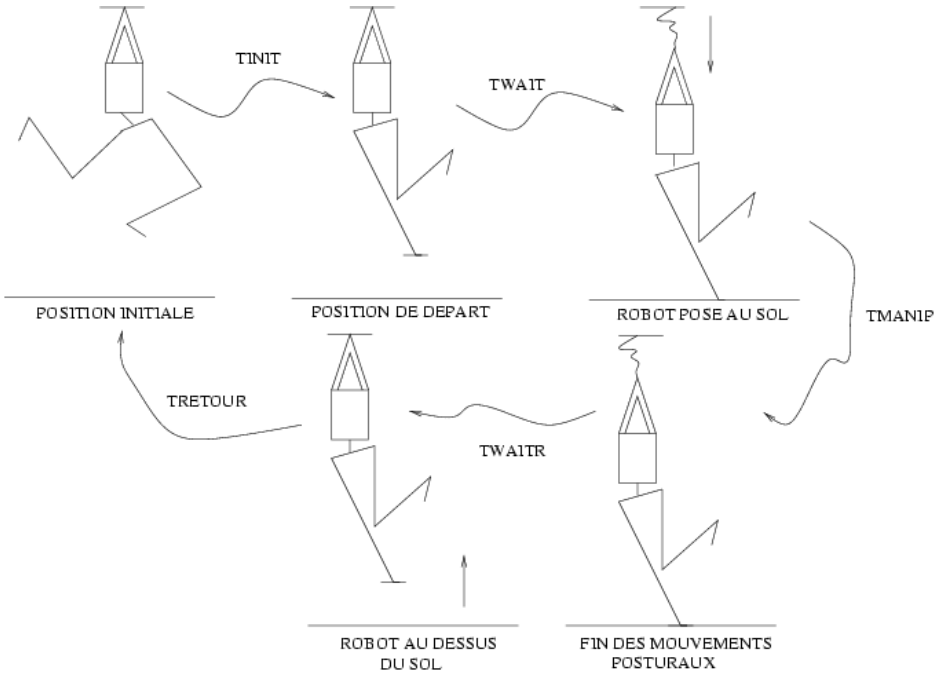
L'analyse a été réalisée avec maple, en affichant à la fois la position désirée et la position réelle du robot, ainsi que l'erreur maximum, pour chaque articulation. Les erreurs obtenues sont inférieures à nos attentes, le fonctionnement du contrôleur dans son ensemble est donc validé.

5 – Trajectoire d'équilibre du robot

5-1 MODE OPERATOIRE :

Le suivi de trajectoire se décompose en 5 étapes :

- le robot est suspendu et atteint la posture de départ du mouvement (position initiale de la trajectoire désirée),
- le robot reste immobile pendant un temps fixé à l'avance pour être posé sur le sol,
- le robot effectue le mouvement désiré,
- le robot reste immobile pendant un temps fixé à l'avance pour être suspendu,
- le robot revient dans sa position initiale.



5-2 VISUALISATION DU CENTRE DE MASSE :

Nous utilisons les valeurs des capteurs de forces pour déterminer la position du centre de masse.

On dispose pour chaque pied de trois valeurs :

- F_T la force exercée par le tibia sur le capteur du talon,
- F_G la force exercée par le tibia sur le capteur avant droit,
- F_D la force exercée par le tibia sur le capteur avant gauche.

Les positions des capteurs de chaque pied sont exprimées dans un repère centré en la projection O de la cheville sur la semelle et d'axes. On notera les coordonnées des capteurs :

- $C_T = (l_T , L_T)$ pour le capteur du talon,
- $C_D = (l_D , L_D)$ pour le capteur avant droit,
- $C_G = (l_G , L_G)$ pour le capteur avant gauche.

Nous faisons l'hypothèse que les semelles se posent toujours à plat sur un sol horizontal afin de simplifier les calculs.

Force de réaction au sol

Nous pouvons calculer la composante verticale F_P de la force de pression qui s'applique sur le pied de support. Elle permet de savoir dans quelle phase de support de la marche on se trouve.

$$F_p = - F_T - F_G - F_D$$

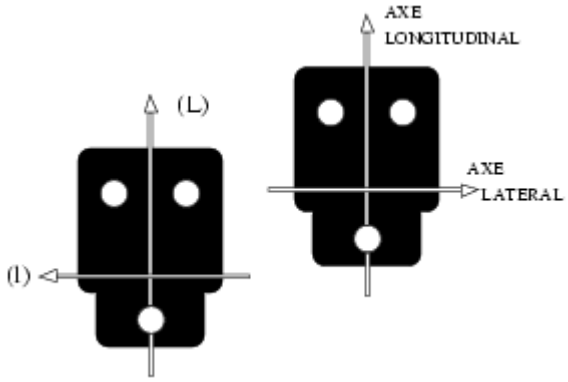
Moment de la force de réaction

On peut également déterminer la projection suivant les axes longitudinaux et latéraux du moment de pression .

$$\begin{cases} M_{pl} = - L_T * F_T - L_G * F_G - L_D * F_D \\ M_{pL} = - l_T * F_T - l_G * F_G - l_D * F_D \end{cases}$$

Centre de pression

Le centre de pression est le point en lequel le moment des efforts de pression est nul.



La position du centre de pression s'obtient aisément :

$$Cp = \begin{pmatrix} C_{pl} \\ C_{pL} \end{pmatrix} = \begin{pmatrix} -M_{pL}/F_P \\ M_{pl}/F_P \end{pmatrix}$$

Nous avons créé un fichier maple qui trace la position du centre de pression en fonction du temps. Ainsi nous pouvons vérifier même si le robot tiens en équilibre s'il est tout de même un peu en avant, un peu en arrière … Ce fichier, ainsi qu'un exemple, est fourni en annexe 4.

5-3 POSTURES D'EQUILIBRE :

a) *Présentation :*

Pour l'étude des postures d'équilibre, nous disposions des trajectoires utilisées sur le robot d'Hanovre. Ces positions ont été calculées par simulation puis validée sur le robot exposé. Il s'agissait pour nous de les tester sur le robot de l'INRIA.

b) *1ier test :*

En testant une des postures d'équilibre sur le pied droit du robot, nous nous sommes aperçus que le robot tombait en arrière. Au vu du comportement du robot, la cause de cette chute ne pouvait pas venir d'un mauvais réglage des PD+Gravité, mais plutôt d'un mauvais positionnement du robot de l'INRIA.

Il s'agit en fait d'un mauvais réglage du zéro mécanique. Les référentiels utilisés dans la méthode précédente ne correspondent pas avec ceux choisis à l'origine. Pour une majorité d'articulation, la première méthode reste valable mais en ce qui concerne le positionnement des hanches et des chevilles dans le plan sagittal, il y a en effet une erreur estimée à environs 6.5 degrés.

c) *Repositionnement du robot :*

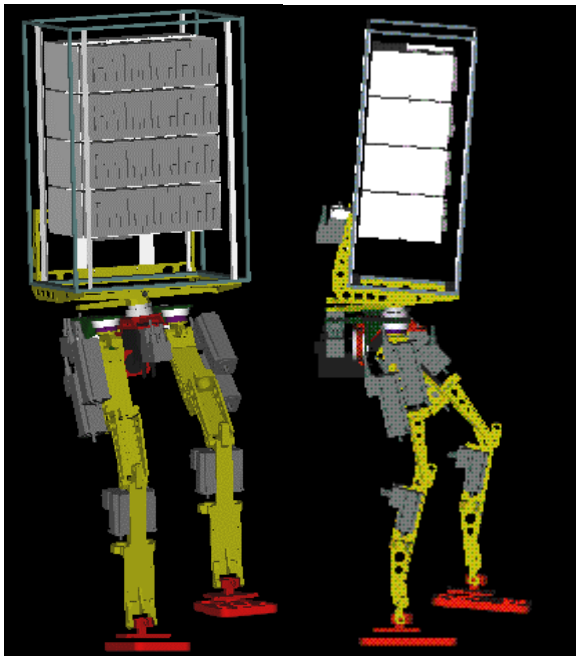
La position de référence consiste en fait à aligner les 3 axes de rotation cheville – genou – hanche sur une même verticale. Du fait qu'il faille conserver le parallélisme entre le tronc et le dessus des pieds, il n'y a donc plus perpendicularité entre le tibia et la cheville.

Nous avons donc repositionné le robot et recalibré les potentiomètres concernes par ce repositionnement. Nous avons également totalement réécrit une méthodologie de positionnement en zéro.

d) *2ieme test :*

Nous avons ensuite rejoué la posture, et en effet le robot s'est maintenu en équilibre. Cette posture fonctionne avec les deux types d'initialisation, c'est-à-dire manuelle ou automatique. L'initialisation automatique n'avait pas été implémentée sur le robot d'Hanovre. Il reste tout de même encore à estimer sa robustesse et sa répétabilité.

Comme énoncé précédemment, nous avons représenté la position du centre de masse. Un exemple de résultat est donné en annexe 4. Nous distinguons bien les phases de pose et de levage. Nous pouvons voir également que le robot est parfaitement en équilibre, le centre de masse se trouvant très proche de l'origine du repère (point où l'équilibre est considéré comme parfait).

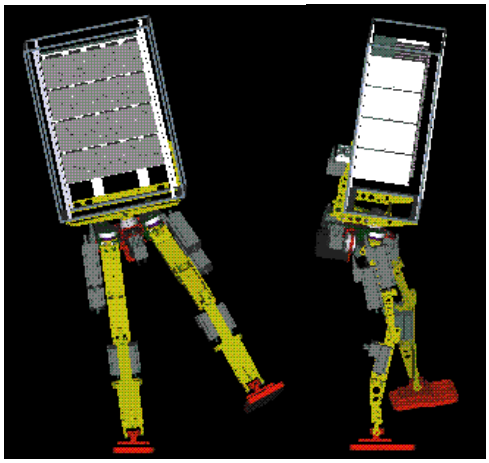


Posture initiale

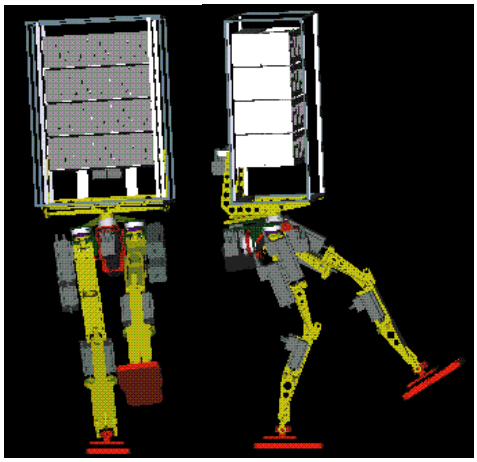
L'étape suivante consistait à tester cette même posture pour le pied gauche. Ceci n'avait jamais été testé auparavant et le résultat a été très concluant.

e) Autres postures :

Nous avons ensuite testé d'autres postures d'équilibre également calculées par simulation et validées sur le robot d'Hanovre. Ces postures sont représentées ci-dessous :



pied grand écart coté



pied en avant

Pour chacune de ces postures, nous avons également contrôlé les positions de chaque articulation et visualisé les erreurs de positions. Certes, il réside toujours une erreur mais elle reste très faible (inférieure à nos attentes).

L'étape suivante qui n'a pu être testée consiste à réaliser un « stand up » du robot (appui sur les 2 pieds).

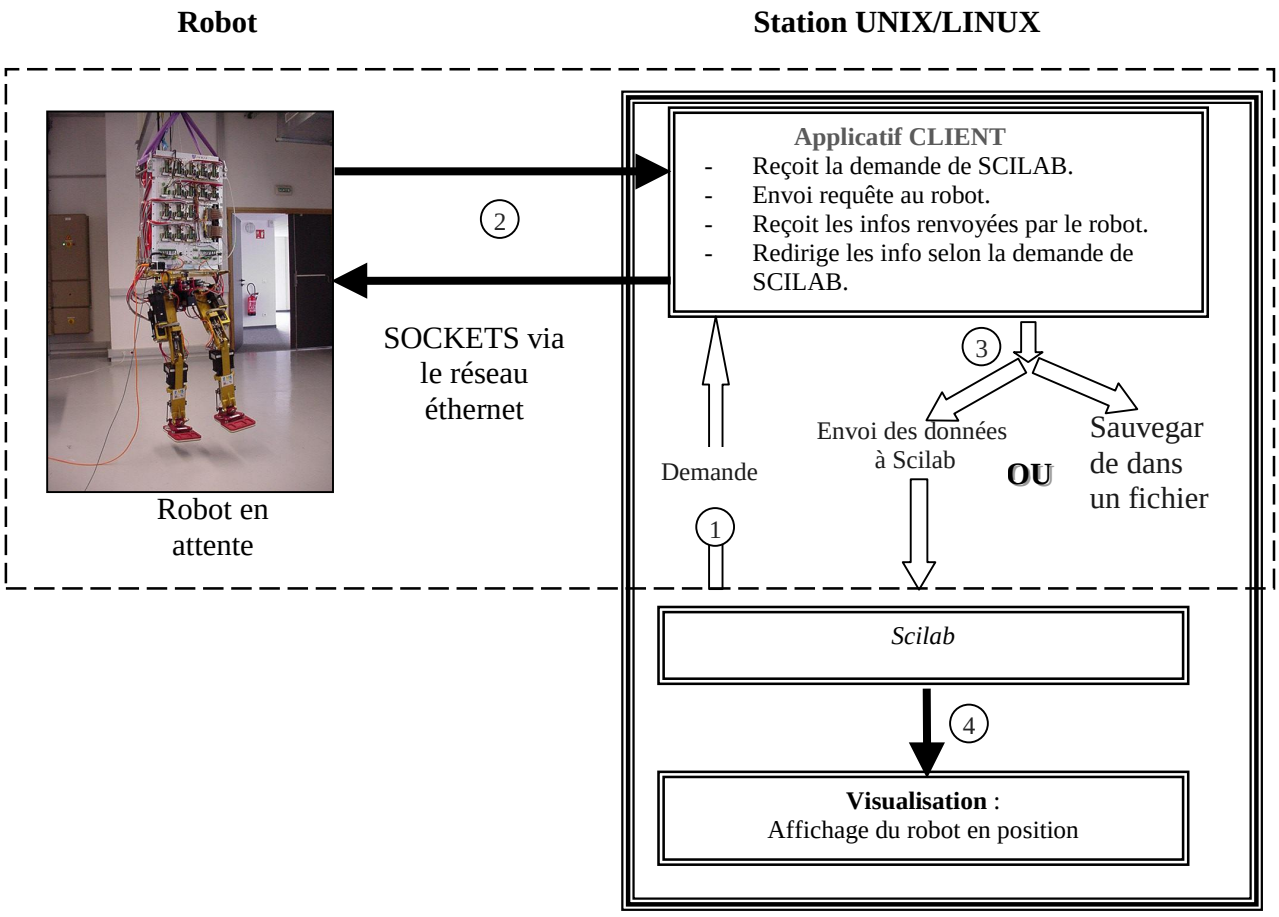
6 - Visualisation

6-1 INTRODUCTION :

En parallèle, j'ai travaillé avec un autre stagiaire pour réaliser la visualisation du robot dans l'espace. Mon rôle consistait à récupérer les informations angulaires de chaque articulation à partir du réseau éthernet (liaison reliant le robot bipède à la station de travail).

La visualisation du robot est établie à partir du logiciel Scilab, c'est donc cet applicatif qui est maître et gère la demande d information. Le robot est utilisé comme SERVEUR, c'est-à-dire qu'il est continuellement en attente, alors que la station unix est considérée comme CLIENT, c'est elle qui demande au robot de lui communiquer les informations. Ce type de communication est appelé SOCKET.

6-2 SCHEMA DE LA COMMUNICATION :



6-3 CHEMINEMENT DE L'INFORMATION :

① SCILAB établit une communication avec l'applicatif dit CLIENT (programme C sous station Unix) et envoie un message qui demande à l'applicatif soit de lui renvoyer les informations robot soit de les sauvegarder dans un fichier.

② Une fois ce message reçu, l'applicatif envoie une requête au robot, celui-ci étant préalablement mis en attente. Le système d'exploitation VxWorks du robot, récupère les informations et les transmet à l'applicatif.

③ L'applicatif, selon le message de SCILAB, renvoie les données à SCILAB ou les sauvegarde dans un fichier.

④ SCILAB ayant récupéré toutes ces informations, lance un programme qui affiche le robot dans l'espace.

6-4 RESULTATS :

La partie concernant la communication entre l'applicatif CLIENT et le Robot marche parfaitement, il reste encore à implémenter la partie qui gère la communication entre Scilab et l'applicatif (pour l'instant des problèmes de version de Scilab génère des problèmes).

Nous avons validé la partie CLIENT / SERVER, en n'utilisant pas la communication entre Scilab et l'applicatif CLIENT. Nous avons mis le robot en attente, et depuis la station Unix nous avons lancé l'applicatif sans passer par Scilab, et en sauvegardant les informations renvoyées par le robot dans un fichier. Nous avons donc obtenu un vecteur à 15 éléments, que nous avons transmis à Scilab manuellement. Ensuite nous avons lancé la visualisation, et en effet le robot représenté correspondait bien à la position réelle.

7 – Partie économique

7-1 RAISONS DU SUJET :

L'intérêt des robots bipèdes réside dans leur capacité naturelle à évoluer dans les environnements de notre vie quotidienne, essentiellement conçus pour la bipédie. Ainsi, la classe d'applications potentielles visée est-elle la robotique de service.

Par ailleurs le robot actuellement utilisé dans l'équipe est conçu comme une plate-forme d'accueil pour des recherches en mécanique, automatique et informatique temps réel.

7-2 INVESTISSEMENT CONSACRE A CE PROJET :

a) Investissement de l'entreprise :

Le coût de l'investissement depuis le début du projet est estime à 1400 KF (2 robots à 700 KF) du point de vue matériel. Du fait qu'il s'agisse d'un projet de recherche, l'amortissement est considéré comme nul.

Du point de vue du coût de travail, le nombre d'ingénieurs intervenant sur le projet est estimé à 4 hommes par an depuis 1996.

L'exacte valeur du coût de travail horaire de l'ingénieur dit « habillé » est de 700 KF par an par ingénieur.

b) Coût de mon travail :

Les indemnités que je perçus sont de l'ordre de 30 kF. (6 mois à 5 KF, indemnités et charges sociales).

7-3 MOYENS MIS EN ŒUVRE POUR LA VENTE EVENTUELLE DU PRODUIT :

- **Présentation publique du robot :**

Le robot a fait l'objet d'une démonstration publique à l'Exposition universelle d'Hanovre (film, photos ..) du 1er juin au 31 octobre 2000.

➪ **Les lieux de démonstration**

La mise en œuvre du robot BIP 2000 en deux exemplaires permet sa démonstration dans les deux sites distants de conception. D'une part, comme prévu initialement dans le projet, Poitiers et le site du Futuroscope sont concernés. En effet, une bonne part des financements existants provient de la région Poitou Charentes et du Conseil Général de la Vienne. D'autre part, Grenoble est aussi simultanément l'autre site de démonstration. Les recherches du projet en contrôle/commande y sont en effet conduites à l'INRIA. Grenoble est la patrie de Vaucanson, et la présentation de ce robot peut être vue comme un lien établi au-delà des siècles avec le précurseur des mécanismes automatiques.

➪ **Publications scientifiques :**

Le robot Bip2000 a fait l'objet de plusieurs publications scientifiques.

➪ **Brevet :**

Brevet Français n° 00 05 169, "Actionneur conçu pour des structures articulées, telles que des membres de robot marcheur", enregistré à l'INPI le 21 avril 2000, P. Sardain, CNRS, France (22 pages).

➪ **BIP2000 et les médias :**

- ARTE, est la chaîne de télévision officielle qui couvre l'Exposition Universelle et en particulier le Pavillon Français.
- Le mercredi 2 juin 1999, la chronique scientifique de Marie-Odile Monchicourt sur France-Info a été consacrée au projet BIP-2000.

7-4 PRINCIPALES ETAPES DU DEROULEMENT DU PROJET :

Le projet a débuté par de la recherche sur la modélisation du robot, sur la simulation et sur l'outil de développement de loi de commande ORCCAD. Cette étape a été réalisée par des doctorants chercheurs.

Pour la partie réalisation du robot, deux prototypes ont été conçus et réalisés par le LMS de Poitiers. Le LMS s'est inspiré des données cinématiques et des capacités dynamiques anthropomorphes pour la conception du robot.

Le calendrier de la réalisation mécanique des deux prototypes a été le suivant :

- Juillet 1999 : construction d'une première version comportant deux jambes (8 ddl) associées à un chariot à roues.
- Janvier 2000 : construction d'un prototype comportant les jambes et le bassin (15 ddl).
- Septembre 2000 : rajout d'un bassin au premier prototype et obtention d'un second prototype à 15 ddl .

L'INRIA avait la responsabilité de la réalisation de l'armoire de commande et des câblages, ainsi que du développement d'un contrôleur pour le robot. Le Service Moyens Robotiques Vision et Réalité Virtuelle de l'INRIA Rhône-Alpes a la charge de la partie électronique du robot. Ce travail correspond au câblage des différents moteurs et capteurs, ainsi qu'à la réalisation de l'armoire de commande et au développement des drivers. Le projet Bip de l'INRIA Rhône-Alpes a la responsabilité de la partie contrôleur du robot. Ce travail a permis de développer la partie logicielle du système et d'implémenter une loi de commande permettant de poursuivre des trajectoires articulaires.

Une fois les prototypes réalisés, une étape de mise au point et de démonstration (notamment des réglages de paramètres, des modifications à apporter…) a été effectuée par des ingénieurs et des doctorants.

Ci-dessous est exposé un calendrier des étapes de conception et d'expérimentations :

- Achèvement des essais sur la jambe test : avril 1998
- Dossier de fabrication de la jambe prototype à 4 degrés de liberté : juin 1998
- Schéma de câblage : septembre 1998
- Réalisation d'un pied muni de capteurs de pression : octobre 1998
- Réalisation de 4 exemplaires de la jambe prototype : octobre 1998
- Montage, livraison, premiers essais à Poitiers et Grenoble des deux ensembles : avril 1999
- Achèvement de la conception du pelvis à 2 x 2 articulations : février 1999
- Réalisation du pelvis : avril 1999
- Montage pelvis/jambes/tronc puis tests : octobre 1999
- Ecriture et test des drivers des moteurs : juin 1999 à janvier 2000
- Marche dans le plan sagittal de l'ensemble sans pelvis articulé BIP1 : mars 2000
- Etude expérimentale de lois de commande simple pour le système complet : avril 2000
- Premiers pas du système complet dans le plan sagittal : décembre 2000
- Premiers pas en 3D du système complet : mai 2001

8 – Perspectives et conclusions

8-1 PROBLEMES RENCONTRES :

Le principal problème rencontré, comme nous l'avons explicité dans ce rapport, a été le mauvais positionnement du robot en zéro mécanique. Le référentiel utilisé en premier lieu ne coïncidait pas avec le référentiel utilisé à l'origine. Il a fallu modifier la première méthodologie pour la rendre conforme et refaire une calibration de certains potentiomètres.

Nous avons également rencontré quelques problèmes pour poser le robot lors des phases d'équilibre. La semelle doit être posée bien à plat et relativement rapidement sinon le pied entre en vibrations.

Des problèmes avec le module de lecture analogique relié aux capteurs de forces nous ont également ralentis. Pour des raisons inexpliquées (pas de modifications de câblage), ce module tombe en panne au bout d'un certain temps. Nous avons donc changé la carte plusieurs fois. Le pourquoi du problème n'a pas été détecté. Les cartes endommagées ont été renvoyées au constructeur.

8-2 PERSPECTIVES :

Dans un avenir proche (fin 2001), les objectifs à atteindre sont un « stand up » du robot, et une marche statique. Le « stand up » consiste à positionner les jambes du robot de la même façon pour que lorsqu'il est posé, il y ait les 2 pieds en contact avec le sol. Ensuite le robot est amené à se lever. La stabilité statique consiste à maintenir la projection verticale du centre de masse du robot à l'intérieur de la surface de sustentation. La marche statique peut être ainsi assimilée à une succession de postures stables. Généralement ce genre de mouvement s'effectue à une vitesse assez réduite pour diminuer l'effet de l'inertie.

L'année 2002 devrait être consacrée à l'étude d'une loi de commande qui permette une marche dynamique du robot. Cette marche correspond à la marche de type humaine.

A plus long terme (2003 à 2005), une deuxième version du robot est envisagée, notamment avec une réduction de poids, une diminution du volume de la baie et une augmentation de la puissance de calcul.

8-3 CONCLUSIONS :

a) Conclusions générales :

Ce stage de six mois dans un laboratoire de recherche de l'INRIA m'a permis de me familiariser avec le monde de la recherche. Contribuer à faire marcher un robot bipède est un sujet passionnant.

Un tel projet exige la collaboration de plusieurs équipes spécialisées dans des domaines aussi différents que la mécanique, l'électronique, l'informatique temps réel et l'automatique. Une telle variété est enrichissante, à la fois sur le plan scientifique en abordant l'intéraction entre les sciences, et sur le plan humain, avec la nécessité du travail en équipe et les difficultés de gestion d'un projet de grande envergure.

J'ai pu me rendre compte de l'importance de la communication. L'aspect relationnel est primordial pour la bonne continuité d'un projet.

Le fait d'effectuer des manipulations sur le robot a rendu ce stage très attrayant. De plus, nous avons bien pu nous rendre compte de l'évolution du travail au fur et à mesure, ce qui est très motivant.

b) Bilan de travail réalisé :

Je présente ci-dessous un bilan succinct de mon travail :

- La synthèse des différents documents sur le robot bipède,
- Rédaction d'une méthodologie pour la mise en zéro mécanique du robot,
- Calibration des potentiomètres (programmes + manipulations),
- Validation de leur utilisation (programmes + analyse des résultats),
- Mise à jour des valeurs butées (programmes + manipulations),
- Etude du modèle et du contrôleur implémenté,
- Elaboration d'une méthodologie qui valide le contrôleur,
- Réglage et mesures de paramètres (manipulations + analyse de résultats),
- Postures d'équilibre du robot (manipulations).
- Visualisation du robot (programmes + test de validité).

ANNEXE 1 : PROGRAMME UTILISE POUR LA RECHERCHE DES BUTEES ELECTRIQUES

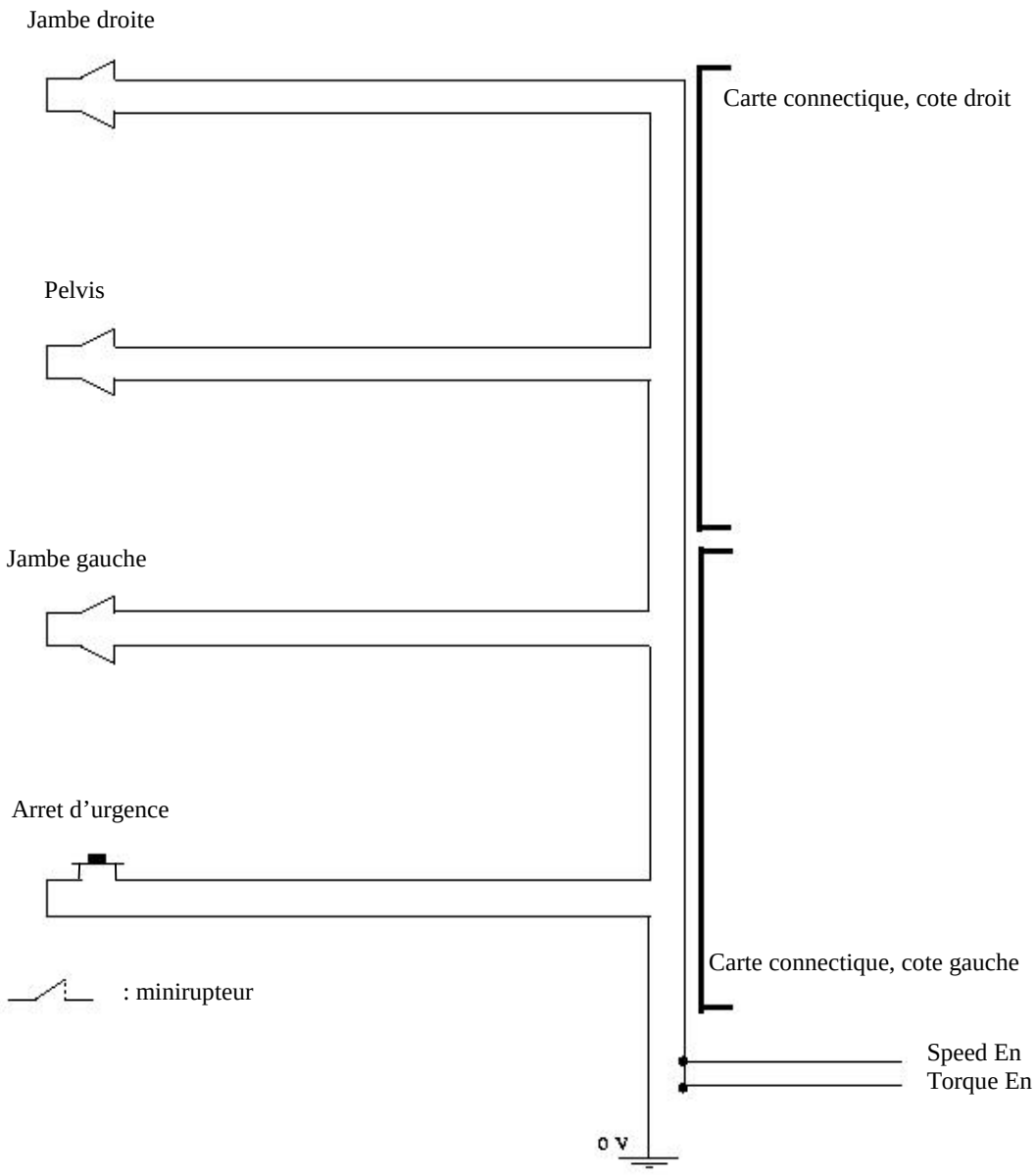
```
/*-------------------------------------------------------------------------------------------*/
/*                          Recherche des Butees electriques                                 */
/*-------------------------------------------------------------------------------------------*/

#include <BipPotars.h>
#include <stdio.h>
#include <math.h>

#ifdef VXWORKS
#include <taskLib.h>
#include <tickLib.h>
#include <sysLib.h>
#include <semLib.h>
#endif

#include "hardBip.h"
#include "utilBip15.h"

                        /*-----------------------------------------*/
                        /*          Definition des variables       */
                        /*-----------------------------------------*/
#define BIP_DDL 15
#define NbCircuitMinirupteur 3

/*POTAR : tableau de passage des entrees module_ip aux entrees bipede*/
static int bipLectPotarCor[BIP_DDL] = {3, 11, 12, 4, 1, 9, 10, 2, 7, 6,
                                       5, 13, 14, 8, 16};

/*tableau des valeurs des potars*/
static double bipLectAdc[BIP_DDL];

/*tableau des valeurs absolues des articulations*/
static double bipPosAbs[BIP_DDL];

/*variable gerant l'initialisation*/
static int bipTestPosAbsInitFlag = ERROR;

/*tableau d etat des minirupteurs*/
static int bipLectDigit[NbCircuitMinirupteur];

            /*-----------------------------------------------------------------*/
            /*    fonction s arrete des qu une butee elect est atteinte        */
            /*   affiche les positions des articulations appartenant au        */
            /*                    circuit en butee                             */
            /*-----------------------------------------------------------------*/
void  bipRechercheButeeElectrique()
{
  /*declaration variable*/
  int bipLectAxe;
```

```
int attentebutee=1;
float ConvRad_Deg=180/3.14159;

/*tableau des coeff directeurs*/
double Pot[BIP_DDL]={   PotCheD_F, PotCheD_S, PotGenD_S, PotHanD_S,
                        PotCheG_F, PotCheG_S, PotGenG_S, PotHanG_S,
                        PotHanD_F, PotHanD_V, PotHanG_F, PotHanG_V,
                        PotLomb_V, PotLomb_F, PotLomb_S};

/*tableau des offsets*/
double Off[BIP_DDL]={   OffCheD_F, OffCheD_S, OffGenD_S, OffHanD_S,
                        OffCheG_F, OffCheG_S, OffGenG_S, OffHanG_S,
                        OffHanD_F, OffHanD_V, OffHanG_F, OffHanG_V,
                        OffLomb_V, OffLomb_F, OffLomb_S};

/*gestion de l'init*/
if (bipTestPosAbsInitFlag == ERROR)
  {
    bipHardInit();
    bipTestPosAbsInitFlag = OK;
  }
/*attente d'une butee elect*/
while (attentebutee==1)
  {
    /*************************************************************/
    /*                  Butee jambe droite                       */
    /*************************************************************/
    if (bipHardGetDigit(2, &bipLectDigit[0]) != OK)
          {
           fprintf(stderr, "bipHardGetDigit retourne ERROR\n");
          }
    else
          {
           if(bipLectDigit[0]==1)
             {
              printf("Un des element de la jambe droite est en butee elect\n");
              /*lecture et affichage des potars correspondants*/
              for(bipLectAxe = 0; bipLectAxe < 4; bipLectAxe++)
                   {
                    if (bipHardGetAnalog(bipLectPotarCor[bipLectAxe]+15,
                                         &bipLectAdc[bipLectAxe]) != OK)
                      {
                       fprintf(stderr, "bipHardGetAnalog retourne ERROR\n");
                      }

/*calcul et affichage des positions absolues de l'articulation en butee elect*/
                    bipPosAbs[bipLectAxe]=(bipLectAdc[bipLectAxe]*Pot[bipLectAxe]
                                           +Off[bipLectAxe])*ConvRad_Deg;
                    printf("Position angulaire de l'articulation %d : %f\n",
                            bipLectAxe, bipPosAbs[bipLectAxe]);
                   }
              attentebutee=0;
             }
          }
    /*************************************************************/
```

```
    /*                  butee pelvis                                                */
    /************************************************************/
    if (bipHardGetDigit(3, &bipLectDigit[1]) != OK)
        {
         fprintf(stderr, "bipHardGetDigit retourne ERROR\n");
        }
    else
        {
         if(bipLectDigit[1]==1)
           {
            printf("Un element du pelvis est en butee elect\n");
            /*lecture et affichage des potars correspondants*/
            for(bipLectAxe = 8; bipLectAxe < 15; bipLectAxe++)
                {
                 if (bipHardGetAnalog(bipLectPotarCor[bipLectAxe]+15,
                                      &bipLectAdc[bipLectAxe]) != OK)
                   {
                    fprintf(stderr, "bipHardGetAnalog retourne ERROR\n");
                   }
/*calcul et affichage des positions absolues de l'articulation en butee elect*/
                 bipPosAbs[bipLectAxe]=(bipLectAdc[bipLectAxe]*Pot[bipLectAxe]
                                        +Off[bipLectAxe])*ConvRad_Deg;
                 printf("Position angulaire de l'articulation %d : %f\n",
                        bipLectAxe, bipPosAbs[bipLectAxe]);
                }
            attentebutee=0;
           }
        }
    /************************************************************/
    /*              butee jambe gauche                                          */
    /************************************************************/
    if (bipHardGetDigit(5, &bipLectDigit[2]) != OK)
        {
         fprintf(stderr, "bipHardGetDigit retourne ERROR\n");
        }
    else
        {
         if(bipLectDigit[2]==1)
           {
            printf("Un element de la jambe gauche est en butee elect\n");
            /*lecture et affichage des potars correspondants*/
            for(bipLectAxe = 4; bipLectAxe < 8; bipLectAxe++)
                {
                 if (bipHardGetAnalog(bipLectPotarCor[bipLectAxe]+15,
                                      &bipLectAdc[bipLectAxe]) != OK)
                   {
                    fprintf(stderr, "bipHardGetAnalog retourne ERROR\n");
                   }

/*calcul et affichage des positions absolues de l'articulation en butee elect*/
                 bipPosAbs[bipLectAxe]=(bipLectAdc[bipLectAxe]*Pot[bipLectAxe]
                                        +Off[bipLectAxe])*ConvRad_Deg;
                 printf("Position angulaire de l'articulation %d : %f\n",
                        bipLectAxe, bipPosAbs[bipLectAxe]);
                }
```

```
            attentebutee=0;
          }
        }
    }
}
```

Câblage normal des minirupteurs « fin de course » dit Daisy Chain



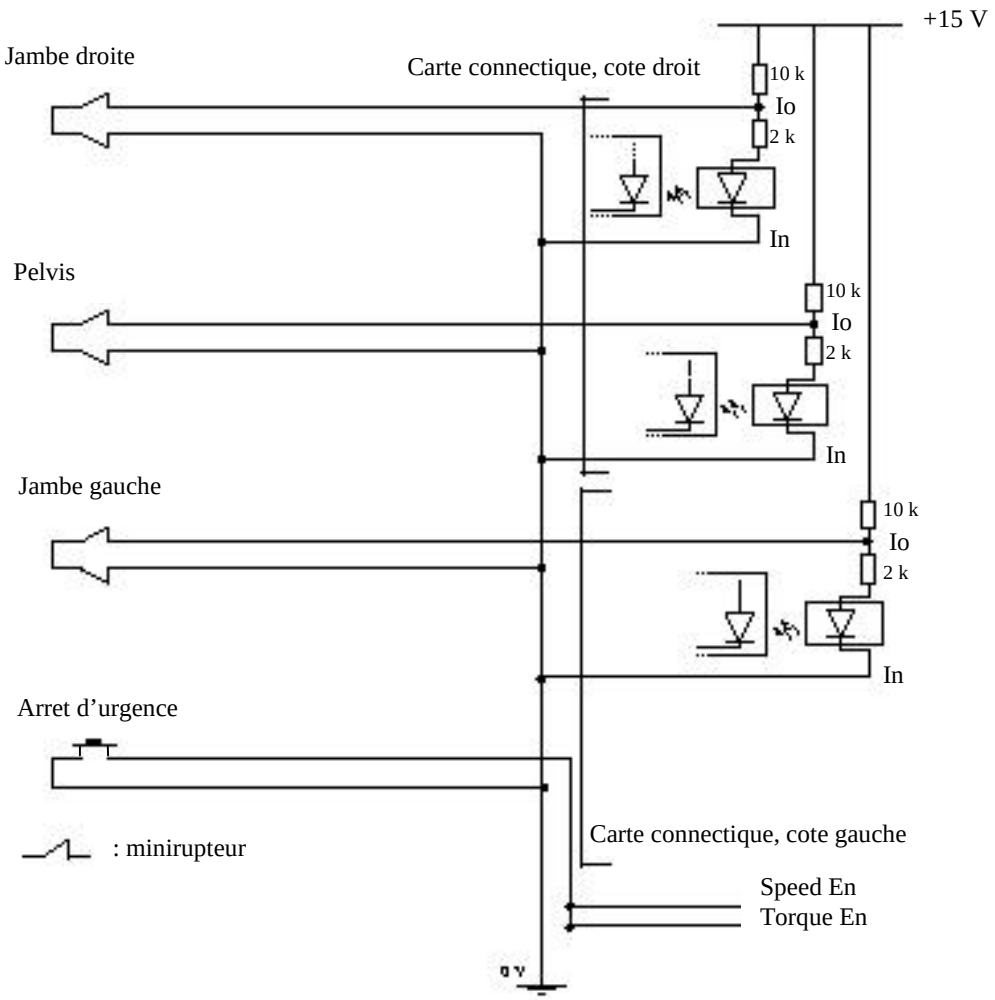
Les minirupteurs sont normalement fermés lorsque l'articulation n'est pas en fin de course.

Circuit fermé : *lorsque aucune articulation n'est en butée électrique*
+
arrêt d'urgence non actionne

Circuit ouvert : *au moins une articulation en butée électrique*
+
arrêt d'urgence actionne

Ainsi, si un fils du circuit se coupe, le circuit s'ouvre et coupe l'alimentation des moteurs.

Câblage des minirupteurs « fin de course » pour permettre la détection des butées électriques :



Les minirupteurs sont normalement fermés lorsque l'articulation n'est pas en fin de course.

Tableau récapitulatif des valeurs butées :

Butées / Articulation	Mécanique Max	Mécanique Min	Electrique Max	Electrique Min	Logicielle Max	Logicielle Min
Cheville Droite Frontal	0,349	-0,349	0,15	-0,17	0,24	-0,24
Cheville Droite Sagittal	0,785	-0,785	-0,77	-0,75	0,68	-0,68
Genou Droit	1,570	0,000	1,49	0,06	1,45	0,10
Hanche Droite Sagittal	0,488	-1,169	0,34	-1,23	0,29	-1,21
Cheville Gauche Frontal	0,349	-0,349	0,17	-0,12	0,24	-0,24
Cheville Gauche Sagittal	0,785	-0,785	-0,83	-0,83	0,68	-0,68
Genou Gauche	1,570	0,000	1,52	0,04	1,45	0,10
Hanche Gauche Sagittal	0,488	-1,169	0,30	-1,24	0,29	-1,21
Hanche Droite Frontal	0,261	-0,261	0,21	-0,20	0,19	-0,19
Hanche Droite vertical	0,261	-0,261	0,19	-0,18	0,15	-0,15
Hanche Gauche Frontal	0,261	-0,261	0,22	-0,24	0,19	-0,19
Hanche Gauche vertical	0,261	-0,261	0,16	-0,19	0,15	-0,15
Lombaire Vertical	0,261	-0,261	0,24	-0,24	0,21	-0,21
Lombaire Frontal	0,174	-0,174	0,15	-0,15	0,09	-0,09
Lombaire Sagittal	0,174	-0,349	0,15	-0,34	0,09	-0,24

Valeurs en radians.

ANNEXE 2 : PROGRAMME UTILISE POUR L'EVALUATION DES FROTTEMENTS SECS.

```
#include <stdio.h>
#include <math.h>
#include <taskLib.h>

#include "hardBip.h"

#define NB_IO 16

typedef struct _FINAL
{
  int cod;
  int adc;
  double couple;
  double adc_mes;
  double adc_zero;
} FINAL;

static FINAL bipResult[NB_IO];

static double codMem[NB_IO];
static double adcMem[NB_IO];

static double cod[NB_IO];
static double adc[NB_IO];

static double adc_zero[NB_IO];

/* REGLAGE A FAIRE : Debut*/
/* nombre d'essai pour le calcul de frottement sec,
   la valeur max sur les moteurs est donc NB_ESSAI * INCR_CONSIGNE
   NB_ESSAI = 8 pour les moteurs a vide
*/
#define NB_ESSAI 80 /*20*/
/* en tick, temps d'attente au demarrage apres consigne moteur,
   pour l'inertie */
#define T_D_MOTEUR 15 /*30*/ /* 60 */
/* en tick, temps d'attente a l'arret apres consigne moteur,
pour l'inertie */
#define T_F_MOTEUR 20
/* nombre de pas codeurs pour valider que le moteur a bouge */
#define DELTA_CODEUR 200.0 /* 300.0 */
/* delta mesure de courant pour valider que le moteur a bouge */
#define DELTA_COURANT 0.1
/* increment sur la consigne de commande */
#define INCR_CONSIGNE ((float)sens * 0.03)  /*((float)sens * 0.06)*/
/* REGLAGE A FAIRE : Fin */

static int bipTestInitFlag = ERROR;   /* variable gerant l'initialisation */

int bipTestCodeur()
```

```
{
  double valeur;
  int i;

  for (i=0;i<10;i++) {
    bipHardGetEncoders(4, &valeur);
    printf("%f\n", valeur);
    taskDelay(30);
  }

  return(OK);
}

int bipTestInit()
{
  if (bipTestInitFlag == ERROR)
    {
      bipHardInit();
      bipTestInitFlag = OK;
    }
  return(OK);
}
/* but : associer les sorties dac aux entrees codeur et adc, plus
   mesurer les frottement sec des moteurs */
static int bipTestMoteurF(int sens)
{
  int m, c, i;
  int fin_test = 0;

  if ((sens != 1) && (sens != -1))
    {
      printf("Pb Argument\n");
      return(-1);
    }

  /* gestion de l'init */
  if (bipTestInitFlag == ERROR)
    {
      bipHardInit();
      bipTestInitFlag = OK;
    }

  /* init */
  printf("init variables\n");
  for (m=0;m<NB_IO;m++)
    {
      bipHardPutAnalog(m, 0.0);
      bipHardGetEncoders(m, &codMem[m]);
      adcMem[m] = adc_zero[m];
      cod[m] = 0.0;
      adc[m] = 0.0;
      bipResult[m].cod = -1;
      bipResult[m].adc = -1;
      bipResult[m].couple = -1.0;
      bipResult[m].adc_mes = -1.0;
```

```
    bipResult[m].adc_zero = -1.0;
   }

  printf("debut charge\n");
  for (m=0;m<NB_IO;m++)
   { /* pour chaque sortie dac */
    printf("dac No %d\n", m);
    fin_test = 0;
    /* increment de sortie: jusqu'a ce que ca bouge */
    for (c=1;(c<NB_ESSAI) && (fin_test==0);c++)
         {
          printf("couple=%.2f\n", INCR_CONSIGNE * (float)c);
          bipHardPutAnalog(m, INCR_CONSIGNE * (double)c);
          taskDelay(T_D_MOTEUR); /* prise en compte de l'inertie moteur */
          for (i=0;i<NB_IO;i++)
           { /* pour chaque entree codeur et adc */
            bipHardGetEncoders(i, &cod[i]);
           }
          bipHardPutAnalog(m, 0.0); /* arret moteur, les donnees sont lues */
          /* pour chaque entree codeur et adc */
          for (i=0;(i<NB_IO) && (fin_test==0);i++)
           {
            if (fabs(codMem[i] - cod[i]) > DELTA_CODEUR)
                 { /* si ca bouge */
                  bipResult[m].cod = i;
                  bipResult[m].couple = INCR_CONSIGNE * (double)c;
                  fin_test = 1;
                 }
           }
          /* prise en compte de l'inertie moteur pour l'arret */
          taskDelay(T_F_MOTEUR);

    /* pour chaque entree codeur et adc: maj Mem */
          for (i=0;i<NB_IO;i++)
           {
            bipHardGetEncoders(i, &codMem[i]);
           }
         }
   }
  for (m=0;m<NB_IO;m++)
   {
    printf("dac No %d / cod No %d / adc no %d / fs= %.2f / cou= %.3f(%.3f)\n",
           m,
           bipResult[m].cod,
           bipResult[m].adc,
           (float)bipResult[m].couple,
           (float)bipResult[m].adc_mes,
           (float)bipResult[m].adc_zero);
   }
  return(0);
}


int bipTestMoteurPos()
{
```

```
 return(bipTestMoteurF(1));
}


int bipTestMoteurNeg()
{
 return(bipTestMoteurF(-1));
}
```

Résultats obtenus :
Résultats du 10 juillet 2001

Correspondance dac / moteur :

Dac No	designation	*No moteur au niveau bipède*
0	MotCheG_I	4
1	MotCheG_E	5
2	MotGenG_S	6
3	MotHanG_S	7
4	MotCheD_E	0
5	MotCheD_I	1
6	MotGenD_S	2
7	MotHanD_S	3
8	MotHanG_F	10
9	MotHanG_V	11
10	MotLomb_V	12
11	MotHanD_F	8
12	MotHanD_V	9
13	MotLomb_D	13
14	MotLomb_G	14
15	X	X

Tension moteur nécessaire pour égaler les frottements secs

No moteur au niv Bipede	designation	bipTestMoteurPos()	bipTestMoteurNeg()
0	MotCheD_E	0.60	-0.03
1	MotCheD_I	0.48	-0.03
2	MotGenD_S	0.53	-0.16
3	MotHanD_S	0.59	-0.37
4	MotCheG_I	0.80	-0.75
5	MotCheG_E	0.66	-0.54
6	MotGenG_S	0.40	-0.03
7	MotHanG_S	0.43	-0.03
8	MotHanD_F	0.91	-0.57
9	MotHanD_V	0.61	-0.35
10	MotHanG_F	0.96	-0.43
11	MotHanG_V	0.63	-0.03
12	MotLomb_V	0.67	-0.66
13	MotLomb_D	1.10	-0.85
14	MotLomb_G	1.8	-0.04

Dans notre application, nous n'avons qu'un seul jeu de frottements secs, c'est pourquoi nous avons réalisé une sorte de moyenne entre bipTestMoteurPos et bipTestMoteurNeg. Le signe est fonction du sens de rotation des moteurs. Les valeurs des frottements secs sont les suivantes :

Variable associée au frottement sec	**Valeur choisie**
Kf0	0.45
Kf1	0.40
Kf2	0.45
Kf3	0.50
Kf4	0.75
Kf5	0.60
Kf6	0.35
Kf7	0.30
Kf8	0.70
Kf9	0.50
Kf10	0.65
Kf11	0.45
Kf12	0.60
Kf13	0.90
Kf14	1.00

ANNEXE 3 : PROGRAMME MAPLE POUR L'EXPLOITATION DES VALEURS NECESSAIRES AU REGLAGE DES PD+GRAVITE

Programme qui affiche les courbes et effectue le calcul de l'erreur max:

```
# Calibration des PID

with(linalg):

# Répertoire contenant les derniéres données mesurées Position desiree et Position reelle
REP:=`/home/ciney/bozonnet/Traj/genouD/1700/`:

print(`Calibration des PID`);

print(`repertoire=`.REP);

print(`Kp genou = 1700`);
fd:=fopen(`/home/ciney/bozonnet/Traj/genouD/1700/p2.data`,READ):
p:=readdata(fd,float):
fclose(fd):
nops(p);

fd:=fopen(`/home/ciney/bozonnet/Traj/genouD/1700/qd2.data`,READ):
qd:=readdata(fd,float):
fclose(fd):
N:=nops(qd);

fd:=fopen(`/home/ciney/bozonnet/Traj/genouD/1700/c2.data`,READ):
c:=readdata(fd,float):
fclose(fd):
nops(c);

#### Trace des 2 courbes
plot([[seq([i,qd[i]],i=1..N-2)], [seq([i,p[i]],i=1..N-2)]], color=[green,blue],
style=[line,line], legend=[Position_desiree, Position_reelle]);

#### Trace du Torque
plot([seq([i,c[i]],i=1..N-2)], legend=[Torque]);

#### Trace de l'erreur
plot([seq([i,qd[i]-p[i]],i=1..N-2)],
legend=[Erreur_entre_les_positions_reelles_et_desirees]);

##### Erreur maximale
print(`Erreur maximale :`);
max(seq(abs(qd[i]-p[i]),i=1..N-2));

print(`FIN`);
```

Sous Maple, on tape la commande ci-dessous, et le tracé s'effectue :

```
> read 'PID.maple';
```

Calibration des PID

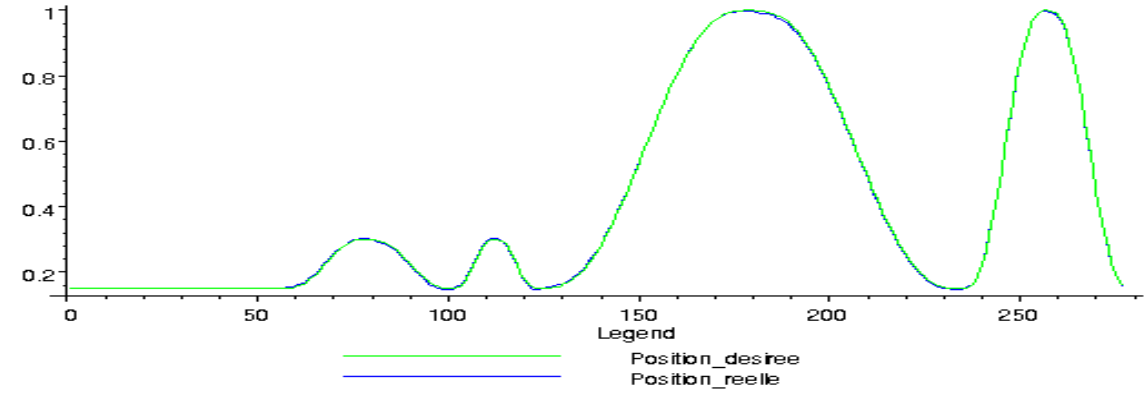
repertoire= . /home/ciney/bozonnet/Traj/genouD/1700/

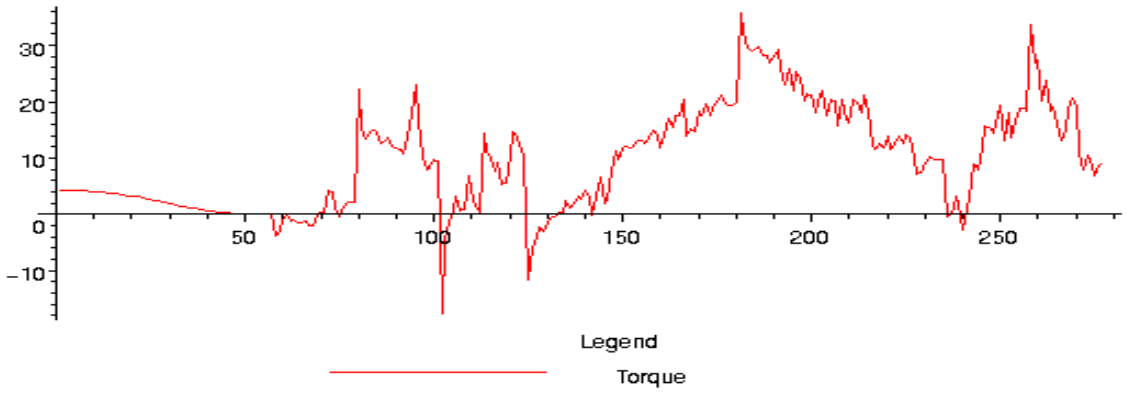
Kp genou = 1700

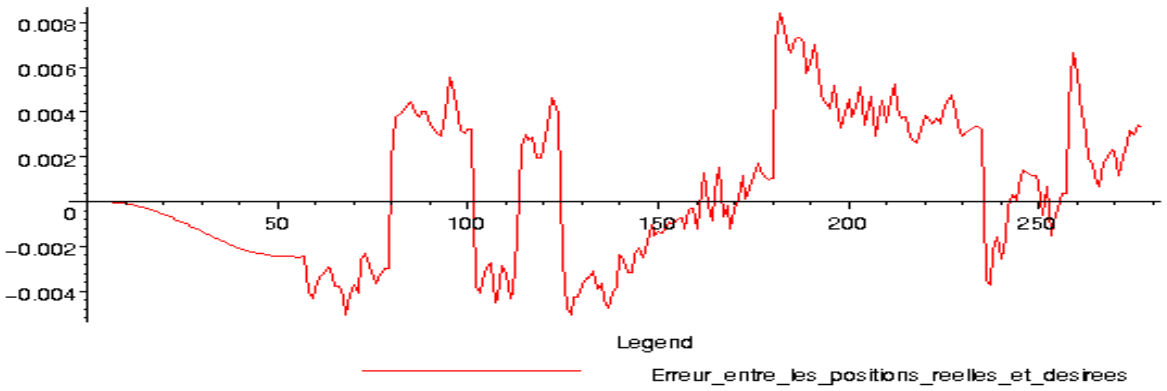
279

N := 279

279







Erreur maximale :

.008411

FIN

ANNEXE 4 : PROGRAMME MAPLE POUR LA VISUALISATION DU CENTRE DE MASSE.

```
#
# Position du centre de masse
#

with(linalg):

print(‘*************************************************************‘);
print(‘Position equilibre sur le pied droit‘);
print(‘*************************************************************‘);

print(‘Capteur effort pied droit‘);

fd:=fopen(‘/home/ciney/bozonnet/Traj/equilibre/f0.data‘,READ):
f0:=readdata(fd,float):
fclose(fd):
N:=nops(f0):
#### Trace du Feet
plot([seq([i,f0[i]],i=1..N-2)], legend=[Feet], color=[green]);
print(‘*************************************************************‘);

fd:=fopen(‘/home/ciney/bozonnet/Traj/equilibre/f1.data‘,READ):
f1:=readdata(fd,float):
fclose(fd):
nops(f1):
#### Trace du Feet
plot([seq([i,f1[i]],i=1..N-2)], legend=[Feet], color=[green]);
print(‘*************************************************************‘);

fd:=fopen(‘/home/ciney/bozonnet/Traj/equilibre/f2.data‘,READ):
f2:=readdata(fd,float):
fclose(fd):
nops(f2):
#### Trace du Feet
plot([seq([i,f2[i]],i=1..N-2)], legend=[Feet], color=[green]);
print(‘*************************************************************‘);


plot([seq([i,18.639 - f0[i] - f1[i] - f2[i]],i=1..N-2)], legend=[force], color=[green]);

plot([seq([i,(18.639 * 0.032 + 0.068 * f2[i] - 0.052 * f1[i] - 0.052 * f0[i])/(18.639 - f0[i]
- f1[i] - f2[i])],i=1..N-2)], legend=[ocp_sag], color=[green]);

plot([seq([i,(0.060 * f1[i] - 0.060 * f0[i])/(18.639 - f0[i] - f1[i] - f2[i])],i=1..N-2)],
legend=[acp_fro], color=[green]);
```

Exemple

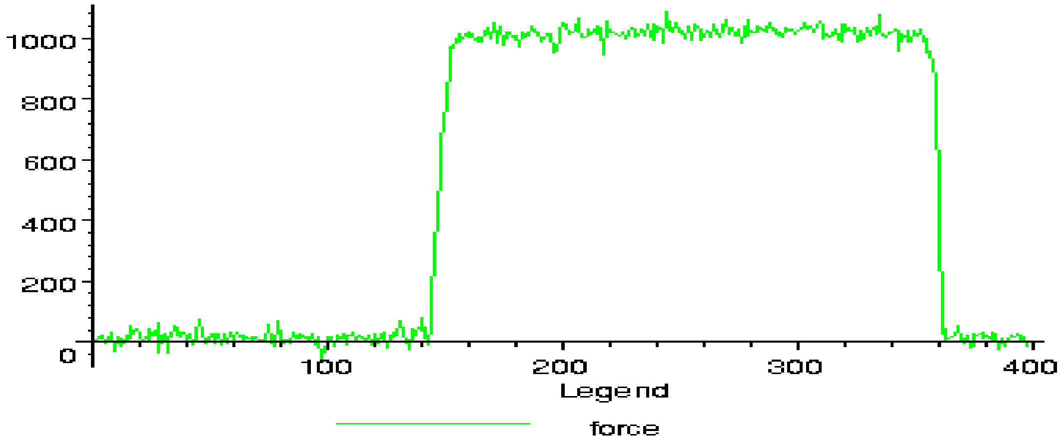
```
> read 'ED.maple';
```

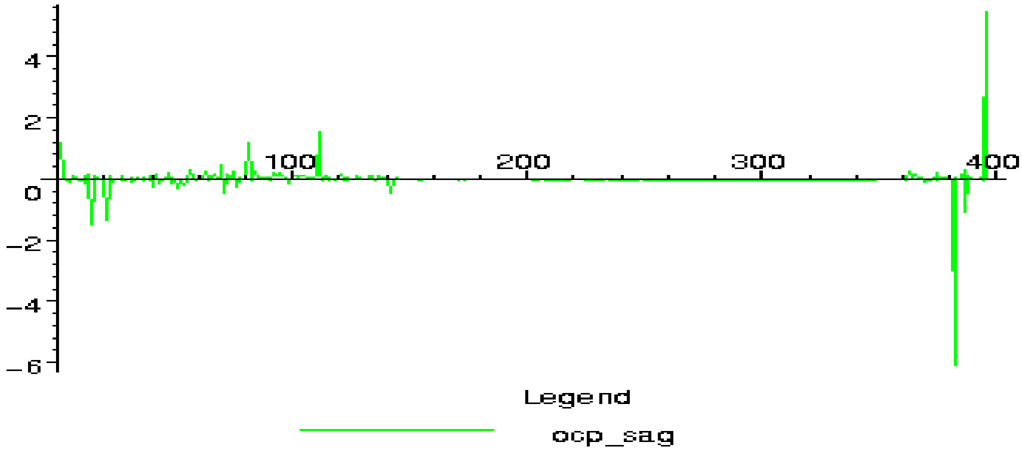
**\\

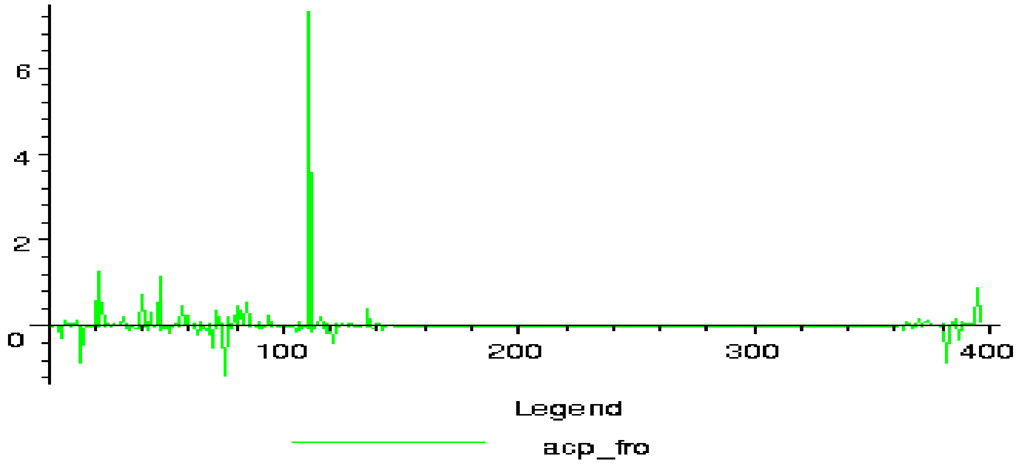
Position equilibre sur le pied droit

**\\

**\\







\>

ANNEXE 5 : PROGRAMMES UTILISES POUR LA VISUALISATION.

Programme chargé sur le robot (VxWorks) :

```
#include <math.h>
#include <stdio.h>

#include "vxWorks.h"
#include "types.h"
#include "taskLib.h"

#include "Rmi.h"
#include "Rsock.h"

#include "hardBip.h"
#include "utilBip15.h"

#include "BipPotars.h"

#define PORT 5001
#define MAX_SIZE_MESSAGE 256
#define BIP_DDL 15

/* variable gerant l'initialisation */
static int bipTestPosAbsInitFlag = ERROR;

/* CODEUR : tableau de passage des entrees module_ip aux entrees bipede */
static int bipLectCodeurCor[BIP_DDL] = {12, 13, 14, 15, 4, 5, 6, 7, 8, 9,
                                        0, 1, 2, 10 ,11};
static double bipLectCodeur[BIP_DDL];
static double bipLectArt[BIP_DDL];

/* POTAR : tableau de passage des entrees module_ip aux entrees bipede */
static int bipLectPotarCor[BIP_DDL] = {3, 11, 12, 4, 1, 9, 10, 2, 7, 6,
                                       5, 13, 14, 8, 16};

static double bipLectAdc[BIP_DDL];
static double bipPosAbs[BIP_DDL];


int bipSockServer(int clientid);

int bipSockServerInit()
{
  if (sockServerOpenMultiClients(PORT, (FUNCPTR)bipSockServer) == ERROR)
    {
    fprintf(stderr,"ERROR : Problem to open socket\n");
    return(ERROR);
    }
  return(OK);
}

int bipSockServer(int clientid)
{
```

```
  SEM_ID semMutex;
  SOCK_SERVER *socket = sockGetStructure(clientid, &semMutex);
  char message[MAX_SIZE_MESSAGE];
  char message1[MAX_SIZE_MESSAGE];
  int bipLectAxe;

/* tableau des coeff directeurs */
  double Pot[BIP_DDL]={ PotCheD_F, PotCheD_S, PotGenD_S, PotHanD_S,
                        PotCheG_F, PotCheG_S, PotGenG_S, PotHanG_S,
                        PotHanD_F, PotHanD_V, PotHanG_F, PotHanG_V,
                        PotLomb_V, PotLomb_F, PotLomb_S};

  /* tableau des offsets */
  double Off[BIP_DDL]={ OffCheD_F, OffCheD_S, OffGenD_S, OffHanD_S,
                        OffCheG_F, OffCheG_S, OffGenG_S, OffHanG_S,
                        OffHanD_F, OffHanD_V, OffHanG_F, OffHanG_V,
                        OffLomb_V, OffLomb_F, OffLomb_S};

  /* SOCKET */
  printf("socket ok\n");

  while (1)
    {
      sockRec(socket, message, MAX_SIZE_MESSAGE);

      semTake(semMutex, WAIT_FOREVER);

      {
          /* gestion de l'init */
          if (bipTestPosAbsInitFlag == ERROR)
            {
              bipHardInit();
              bipTestPosAbsInitFlag = OK;
            }

          /* lire les donnees du robot */
          for(bipLectAxe = 0; bipLectAxe < 15; bipLectAxe++)
            {
              if (bipHardGetAnalog(bipLectPotarCor[bipLectAxe]+15,
                                   &bipLectAdc[bipLectAxe]) != OK)
                {
                    fprintf(stderr, "bipHardGetAnalog retourne ERROR\n");
                }
            }

          /* calcul des positions absolues en radians*/
          for(bipLectAxe = 0; bipLectAxe < 15; bipLectAxe++)
            {
              bipPosAbs[bipLectAxe]=bipLectAdc[bipLectAxe] * Pot[bipLectAxe]
                + Off[bipLectAxe];
            }

          /* mise des donnees robots dans une string */
          sprintf(message,"%f\n%f\n%f\n%f\n%f\n%f\n%f\n%f\n%f\n%f\n%f\n%f\n%f\n
%f\n%f",
```

```
                    bipPosAbs[0], bipPosAbs[1], bipPosAbs[2],
                    bipPosAbs[3], bipPosAbs[4], bipPosAbs[5],
                    bipPosAbs[6], bipPosAbs[7], bipPosAbs[8],
                    bipPosAbs[9], bipPosAbs[10], bipPosAbs[11],
                    bipPosAbs[12], bipPosAbs[13], bipPosAbs[14]);

          /* envoi de la string */
     sockSendString(socket, message);
     }
     semGive(semMutex);
   }
  printf("closing socket\n");
  sockClose(socket);
  return(OK);
}
```

Programme Applicatif lancé sous station UNIX :

```
#include <math.h>
#include <stdio.h>

#include "Rmi.h"

#include "Rsock.h"

#include <stdlib.h>

#define PORT 5001
#define MAX_SIZE_MESSAGE 256

#define CHEMIN "/home/ciney/bozonnet/Mesures2"

static int bipLectSaveFile(char *name, char message[MAX_SIZE_MESSAGE]);

static char message[MAX_SIZE_MESSAGE];
static char message1[MAX_SIZE_MESSAGE];

static SOCK_SERVER s;

/* Communications headers */
#include <libCalCom.h>
#include <libCom.h>

extern char *TheAppli;
extern char *TheType;
extern char *TheMsg;
extern void QuitAppli();
extern void EndAppli();
extern void ParseMessage();
extern void MsgError();
extern actions_messages tb_messages[];

int bipSockInit()
{
```

```
  int tempo=0;

  while (sockClientOpen(&s, "194.199.21.105", PORT) != OK)
    {
      sleep(2);

      tempo++;
      if (tempo > 10)
            {
              fprintf(stderr,"sock: Give up...\n");
            }
      else
            {
              fprintf(stderr,"sock: Retrying...\n");
            }
    }
  printf("connection ok\n");

}

int bipSockRec()
{
  if (sockIsAlive(&s) == ERROR)
    {
      printf("problem socket...closing\n");
      sockClose(&s);
      sprintf(message ,"Error");
      return(EXIT_FAILURE);
    }
  else
    {
      sockSendString(&s, "v");
      sockRec(&s, message, MAX_SIZE_MESSAGE);
      printf("message received <%s>\n", message);
      return(EXIT_SUCCESS);
    }
}

int bipSaveFile()
{
  char name[64];

  /* sauvegarde des resultats dans les fichiers */
  /* attribue a name le chemin et nom de fichier de sauvegarde */
  sprintf(name, "%s/visu.data", CHEMIN);
  if (bipLectSaveFile(name, message) == ERROR)
    {
      printf("Probleme de generation de fichier\n");/*message d'erreur */
      return(EXIT_FAILURE);
    }
  return(EXIT_SUCCESS);
}

int bipSendScilab()
{
```

```
  printf("BipSendScilab\n");
  /* Loop waiting for messages */
  while (1) {
    scanner_messages();
    if (TheType != NULL) {
      printf("Message received from %s\n",TheAppli);
      printf("    type: %s\n",TheType);
      printf("    message: %s\n",TheMsg);
      if(!strcmp(TheMsg,"envoie")) break;
      TheAppli = NULL; TheType = NULL;TheMsg = NULL;
    }
  }

  bipSockRec();

  printf("avant envoi\n");
  envoyer_message_parametres_var(ID_GeCI,
                                 MSG_POSTER_LISTE_ELMNT,
                                 "type",
                                 message,
                                 NULL);
  printf("apres envoi\n");
  return(EXIT_SUCCESS);
}

int bipSockClose()
{
  printf("closing socket\n");
  sockClose(&s);
  return(OK);
}

int usage()
{
  printf("Usage : bipSockUnix [-write]\n");
  printf("-write : write datas into file visu.data\n");
  printf("default is to send data to scilab\n");
  return(EXIT_SUCCESS);
}

int main(int argc, char **argv)
{
  int igeci;
  int p1, p2;
  char myhost[128];

  int write = 0;

  if(argc >2)
    usage();
  else
    if(argc == 2 && !strcmp(argv[1],"-write"))
      write = 1;

  bipSockInit();
```

```
  if(write){
    printf("write 1");
    bipSockRec();
    bipSaveFile();
    bipSockClose();
    printf("fin");
    return(OK);
  }
  else{

    igeci = find("-pipes",argc,argv);
    if (igeci == -1) exit(1);

    p1 = atoi(argv[igeci+1]); p2 = atoi(argv[igeci+2]);

    /* Intialization of communications */
    init_messages(tb_messages,p1,p2);

    /* Get the name of my computer */
    gethostname(myhost,128);

    while(1)
      bipSendScilab();
  }
}

/*-------------------------------------------------*/
/*      fonction de sauvegarde des valeurs     */
/*-------------------------------------------------*/
static int bipLectSaveFile(char *fname, char *message)
{
  FILE *fp;

  /* open the file */
  fp = fopen(fname,"w");
  if (!fp) return ERROR;

  /* sauvegarde */
  fprintf(fp, "%s\n", message);

  /* fermeture du fichier */
  fclose(fp);

  return(OK);
}
```

BIBLIOGRAPHIE.

[1] Gérard Baille - Pascal Di Giacomo – Hervé Mathieu – Roger Pissard-Gibollet : « L'armoire de commande du robot bipède bip2000 ». Rapport technique, juillet 2000, INRIA Rhône-Alpes.

[2] Christine Azevedo - Roger Pissard-Gibollet : «Le contrôleur du robot Bip2000 ». Rapport technique, avril 2001, INRIA Rhône-Alpes.

[3] *http://www.inrialpes.fr/iramr/* : le service robotique de l'INRIA Rhône-Alpes.

[4] *http://www.inrialpes.fr/iramr/ private/Bipede* : notes techniques sur le robot.

[5] *http://www.inrialpes.fr/iramr/ Orccad* : le logiciel de contrôle-commande ORCCAD.

[6] *http://www.inrialpes.fr/bip/Bip-2000/index-fr.html* : le site public sur la réalisation du robot bipède.