



Formations initiales LÉONARD DE VINCI

Ecole Supérieure d'Ingénieurs Léonard de Vinci

Maquette Robotisée de spectromètres à neutrons

Etudiant : Jean-Philippe Panaget

Option : CS

Promotion : S07

Semestre : 7

Entreprise : Institut Laue Langevin

Maître de Mission : Emmanuel Farhi

Tuteur ESILV : Rémy Sart

CONFIDENTIALITE DU RAPPORT :

Consultation libre : oui

Confidentiel : ☐

Rapport de Mission



Préambule

Pour ce stage, j'ai souhaité mettre en œuvre les compétences acquises pendant la première année de ma formation d'ingénieur au service d'un organisme de recherche. Suite, à différents entretiens, mon choix s'est porté sur l'Institut Laue Langevin.

Le poste à pourvoir au sein du groupe « Calcul Scientifique » répondait totalement à mes attentes. La mission était de réaliser des modèles de spectromètres neutron en Lego MindStorms couplés à un logiciel de simulation numérique des instruments. Cette annonce m'a tout de suite intéressé en raison du domaine que je ne connaissais pas et donc que j'allais pouvoir découvrir mais également pour la technologie employée afin de réaliser la maquette que j'avais déjà pu découvrir lors d'un précédent projet scolaire et qui m'avait tout à fait séduit.

Dans un premier temps, il me sera confié la tâche de me documenter sur le sujet abordé. Par la suite, je débiterai la conception de la maquette puis sa réalisation en portant une attention particulière à son couplage avec McStas et enfin, je finaliserai le package distribuable contenant la documentation utilisateur.

En conclusion, il m'était proposé de travailler dans les domaines tout à fait en adéquation avec les compétences que j'ai pu acquérir ces dernières années. Le stage m'offrait la possibilité de découvrir des domaines que je n'ai jamais pu rencontrer auparavant, tel que la physique fondamentale. L'ambiance de travail que j'ai pu apprécier lors de mon entretien téléphonique et qui m'a paru tout à fait sympathique m'a également beaucoup influencé dans mon choix d'entreprise.

Sommaire

Préambule.....	3
Sommaire	4
Résumé de la mission.....	6
Executive resume	7
Remerciements.....	8
Table des abréviations	9
Introduction.....	10
L'Institut Laue-Langevin.....	11
Le Polygone scientifique de Grenoble	11
Historique.....	12
ILL, un institut européen.....	12
Le Groupe Calcul Scientifique – Division Science	14
Domaines d'expertise du Neutron	15
Spectromètres.....	16
TOF type IN6.....	16
TAS type IN8	23
Modélisation des spectromètres	25
Cahier des charges	25
Une Solution : Mindstorms NXT	25
Caractéristiques générales.....	25
Modélisation mécanique des instruments	27
Description mécanique du TOF IN6	27
Description mécanique du TAS IN8	28
Modélisation CAO	29
Code.....	34
La version « standalone ».....	34
Utilisation.....	34
La version couplée à McStas	36
Utilisation.....	38
Caractéristiques de la maquette.....	39
Asservissement des moteurs	40
Package final.....	41
Conclusion	42
Annexes.....	43
Index thématique.....	43
Installer NXT et NXC sur Ubuntu 10.04	91
Modèles CAO du nxTAS et nxTOF	94
Commande TAS	107
Commande TOF.....	108
Utiliser les maquettes nxTAS et nxTOF	109
Bibliographie.....	114

Résumé de la mission

Mon stage au sein du groupe **Calcul Scientifique** de l'**Institut Laue Langevin** m'a permis de conduire un projet de l'élaboration du cahier des charges jusqu'à la rédaction de la documentation destinée à l'utilisateur.

Mes premières tâches ont consisté à m'informer sur les instruments de l'Institut Laue Langevin et notamment sur les instruments IN6 et IN8 qui sont deux spectromètres dont j'ai pri exemple pour réaliser mes maquettes de TOF (temps de vol) et TAS (trois axes). Par ailleurs, il m'a été proposé de réaliser la maquette en Lego Mindstorms NXT, technologie que j'ai déjà eu l'occasion de pratiquer et qui permet un large choix en terme de possibilités de programmation. Mon choix s'est vite porté vers un langage se rapprochant du C que j'ai eu l'occasion de découvrir en première année du cycle ingénieur à l'ESILV et qui permet d'utiliser des fonctions avancées.

Une fois la maquette validée mécaniquement par le logiciel de CAO et par différents tests, j'ai entrepris de réaliser le programme embarqué permettant de l'utiliser. En se basant sur deux scénarii possibles, j'ai développé des programmes permettant d'utiliser à la fois la maquette sans ordinateur, avec une IHM mais également en la couplant avec le logiciel de simulation d'instruments neutronique développé en partie par l'institut.

Pour finir, j'ai rédigé la documentation utilisateur permettant à la fois de se procurer la maquette, de la construire, de configurer la connexion entre le dispositif et un ordinateur et de l'utiliser.

Executive resume

I did my internship at the **Institut Laue Langevin** and more specifically in the **Computer for Science** group where I conducted a whole project from the specifications to the writing of the documentation for the user.

My first task was to learn basic knowledge about neutron scattering and the instruments available at the Institut Laue Langevin. The instruments IN6 (time-of-flight) and IN8 (three axis) are two spectrometers that I have used as example to produce mock-up's of TOF and TAS. Moreover, I was proposed to produce the models in Lego Mindstorms NXT, that I already had the opportunity to practice and allows a wide choice in terms of programming possibilities. My choice was quickly directed to a programming language approximating the C that I had the opportunity to experience first-year engineering at ESILV and allows to use advanced features.

Once the model was validated by mechanical CAD software and various tests, I began to assemble the Lego model and the embedded programs to use. Based on two scenarii, I developed programs that use both the model without a computer (with an embedded interface) but also the coupling with the McStas neutron instruments simulation software developed in part by the Institute.

Finally, I wrote the user documentation which allows to get the layout, construct, configure the connection between the device and a computer and use it.

Remerciements

J'aimerais bien évidemment débiter cette page en adressant toute ma reconnaissance à mon maître de stage Emmanuel Farhi pour le temps qu'il m'a consacré, les nombreuses solutions qu'il m'a apporté et enfin pour sa sympathie.

Je remercie également les personnes avec qui j'ai pu échanger des idées ou qui m'ont orienté sur certaines pistes techniques, je pense notamment à Yannick Raoul et plus généralement à l'ensemble des membres du groupe Calcul Scientifique.

Une pensée particulière pour les thésards du groupe avec lesquels j'ai passé de fantastiques pauses cigarettes et soirées grenobloises en tant que fumeur passif, je pense notamment à Bachir Aoun qui soutient mi-décembre et je souhaite à Laura Bédouret de révolutionner la science avec ses clathrates (ainsi que d'être un peu moins mauvaise perdante au baby-foot).

Je souhaite remercier le personnel administratif de l'institut, à commencer Mme Muller qui m'a permis de trouver un logement dès mon arrivée. Je remercie également la secrétaire du groupe Brigitte Dubouloz pour sa bonne humeur et son aide dont j'ai eu besoin à plusieurs reprises.

Pour compléter cette page, je souhaiterais remercier des personnes qui n'auront peut-être jamais l'occasion de lire ce document mais qui ont compté dans ma vie grenobloise. Ainsi, je remercie Alexis, le pizzaiolo du coin de la rue pour sa bonne humeur et son rire communicatif, ainsi que Haziz, épicier place Saint-Bruno (à côté de la laverie) unique revendeur grenoblois de Nutella conditionné en pot de 220g et avec qui j'ai passé des moments mémorables devant le «Canal Football Club Algérien».

Enfin, je remercie les professeurs de l'Ecole Supérieure d'Ingénieurs Léonard de Vinci, et notamment Rémy Sart, mon tuteur pour ce stage.

Table des abréviations

Abréviation	Signification
ILL	Institut Laue-Langevin
TOF	Temps de Vol (Time of Flight)
TAS	Spectromètre Trois Axes (Triple Axis Spectrometer)
ESILV	Ecole Supérieure d'Ingénieurs Léonard de Vinci
CS	Calcul Scientifique
NXC	Not Exactly C
PULV	Pôle Universitaire Léonard de Vinci
IHM	Interface Homme-Machine

Introduction

Pour cette seconde année du cycle ingénieur de l'ESILV (Ecole Supérieure d'Ingénieur Leonard de Vinci), j'ai choisi d'effectuer mon stage dans un institut de recherche européen, l'ILL (Institut Laue Langevin). En corrélation avec mon cursus, j'ai été admis, pour une durée de 5 mois, au sein de la division "Science" et plus spécifiquement, dans le groupe "Calcul Scientifique" sous la supervision d'Emmanuel Farhi.

Mon travail consiste à modéliser des spectromètres à neutrons en utilisant les LEGO NXT Mindstorms, puis à les coupler au logiciel de simulation d'instruments, McStas, dont Emmanuel Farhi est l'un des développeurs. Il s'agira par la suite, de développer une interface IHM et de rédiger un guide utilisateur. Ce stage me permet donc de suivre intégralement le projet de la conception au rendu final.

L'ensemble sera utilisé dans les centres de recherche neutronique à titre de démonstrateur (show room), dans les écoles de formation en diffusion neutronique ou pour les manifestations publiques scientifiques (journées portes ouvertes, Fête de la Science ...).

Dans un premier temps, ce rapport vous présentera l'institut et ses domaines de recherche et en particulier certains des instruments présents sur le site. Par la suite, j'aborderai mon travail de conception, puis de réalisation des maquettes.

L'Institut Laue-Langevin

L'**Institut Laue-Langevin** (ILL), est un organisme de recherche international, leader mondial en sciences et techniques neutroniques, implanté sur le polygone scientifique de Grenoble. Doté de la source de neutrons, la plus intense au monde, l'ILL propose aux scientifiques des pays partenaires une instrumentation associée de haute technologie.

Plus de 750 expériences y sont effectuées chaque année, et environ 1500 chercheurs viennent y réaliser leurs programmes, sélectionnés par un comité scientifique. Les scientifiques de l'ILL offrent leur savoir-faire à la communauté scientifique : chimistes, physiciens, biologistes, spécialistes de magnétisme, ...

Le Polygone scientifique de Grenoble

Le Polygone Scientifique de Grenoble comprend un ensemble de laboratoires de recherches à la fois privées et publiques qui représente près de 10 000 salariés et chercheurs.

On peut citer le CEA, le LETI, le CNRS, l'ESRF, l'EMBL ou encore STMicroelectronics, ST-Ericsson, BioMérieux et Minatec, le centre européen des nanotechnologies et bien sur l'Institut Laue-Langevin.

Historique

L'ILL a été fondé en janvier 1967 sous l'impulsion Franco-allemande. L'objectif était de concevoir une source de neutron dédiée à la recherche civile. En 1971, le premier réacteur à haut flux européen était créé en offrant à la communauté scientifique, la première source de neutrons froids du monde. Le Royaume Uni a rejoint les deux pays principaux en 1973, et depuis six pays ont apporté leur partenariat à l'institut.

Max Von Laue (1879-1960)

Il a inventé une méthode de mesure des longueurs d'onde des rayons X utilisant la diffraction qui est à l'origine de toutes les méthodes d'analyse par diffraction. Pour ce travail, qui a aussi permis une étude fine de la structure des cristaux, il a reçu le prix Nobel de physique.

Paul Langevin (1872-1946)

Physicien français spécialiste du magnétisme, des ultrasons et de la relativité. Ses travaux sur la modulation des neutrons rapides ont été déterminants pour la réalisation des premiers réacteurs de recherche.

ILL, un institut européen

Pays fondateurs:

La France avec la participation du Commissariat à l'Energie Atomique (CEA) et du Centre National de la Recherche Scientifique (CNRS).

L'Allemagne avec la Forschungszentrum Julich (FZJ).

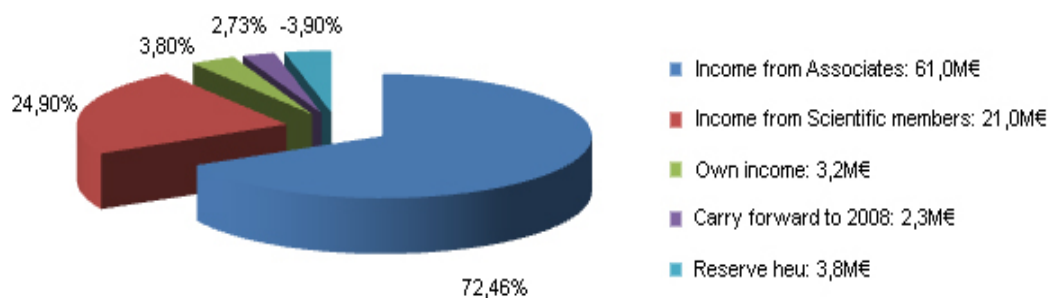
Le Royaume Uni avec le Science and Technology Facilities Council (STFC).

Pays partenaires : Autriche, Espagne, Italie, République Tchèque, Russie, Suisse, Suède, Hongrie, Belgique, Slovaquie, Danemark, Pologne.

ILL en Chiffre

L'ILL emploie environ 500 personnes dont : 90 scientifiques, une trentaine de thésards, plus de 200 techniciens, 65 spécialistes de la sûreté et de l'exploitation, et une cinquantaine d'administratifs. Il y a 64% de Français, 12,5% d'Allemands, 13,5% d'Anglais et environ 10% d'autres nationalités.

Le Budget en 2009 était d'environ 85 ME :



Depuis 2000, un large projet de modernisation des instruments a été lancé et financé à hauteur de 140ME. Il sera achevé en 2014.

Le Groupe Calcul Scientifique – Division Science

Le groupe Calcul Scientifique, dans lequel j'ai été intégré, supporte les scientifiques ILL, étudiants ou visiteurs dans un certain nombre d'activités comme le traitement de données, ou la simulation des instruments et d'échantillons.

Traitement de données

Un certain nombre de programmes écrits en Fortran, Java, Matlab et IDL sont développés et tenus à jour par le groupe .

Simulation d'instruments

Il y a beaucoup d'intérêts à simuler les instruments que ce soit pour les améliorer ou pour optimiser leur conception. Le logiciel mondialement utilisé pour ces simulations est McStas, conjointement développé par le groupe CS et le laboratoire Risoe, Danemark.

Modélisation

Il est possible de simuler et de modéliser les spectres de diffusion de neutrons de molécules de plus en plus complexes grâce aux outils de modélisation couplés au cluster.

MultiMedia

Le groupe est aussi en charge des présentations multimédia mettant en avant l'ILL et permettant la vulgarisation des techniques utilisées à l'institut.

Domaines d'expertise du Neutron

Les neutrons interagissent avec les noyaux des atomes et le spin du neutron interagit avec les moments magnétiques des atomes. Les expériences de diffraction neutronique conduiront à des cartes de densité nucléaire ou à des cartes de densité magnétique.

Magnétisme

Le neutron est l'unique source capable de « voir » en même temps le noyau de l'atome et de « voir » les mécanismes magnétiques de ses électrons.

Matériaux

Le neutron pénètre les matériaux sans les détruire. Il permet notamment de comprendre et d'analyser les contraintes dans un matériau.

Polymères et biosciences

Le neutron peut voir les atomes les plus légers, le neutron est donc idéal pour suivre les mouvements dans des milieux complexes.

Spectromètres

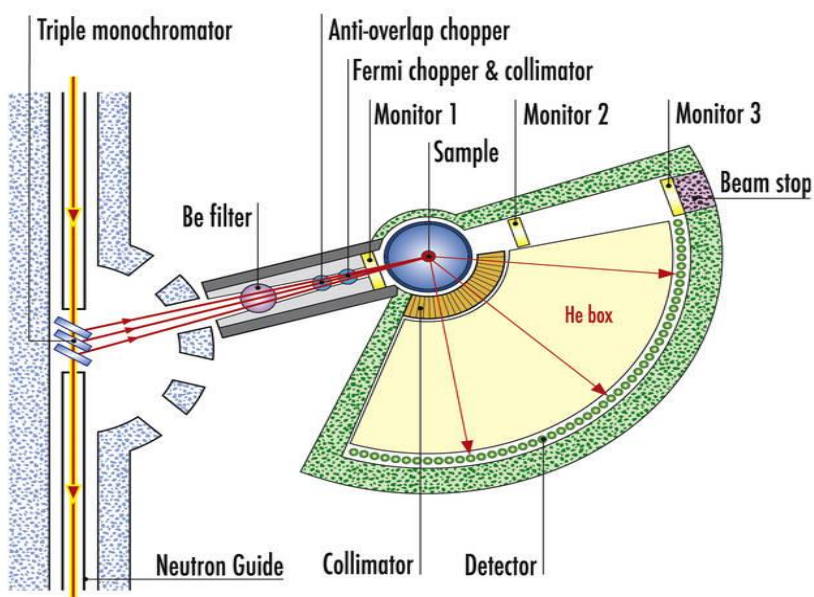
Un spectromètre est un appareil destiné à la mesure de la répartition d'un rayonnement en fonction de la longueur d'onde, de la masse ou de l'énergie de particules individuelles. Concrètement, le spectromètre dirige le flux de neutron sur un échantillon que l'on souhaite étudier, puis détecte la configuration des neutrons à l'arrivée (énergie, position, temps de vol ...). Ces données permettent de décrire les mouvements atomiques et moléculaires de l'échantillon

TOF type IN6

Le spectromètre à temps de vol mesure la position finale du neutron, ainsi que le temps mis pour traverser l'échantillon et arriver jusqu'au détecteur. On connaît la vitesse initiale des neutrons dans le flux car on a fait une sélection par longueur d'onde des neutrons à l'aide d'un monochromateur. De plus, pour connaître la position initiale du neutron, on utilise des choppers qui ne laissent passer qu'une partie du flux à un instant précis (on dit qu'on "pulse" le flux).

On connaît donc la position et la vitesse initiale des neutrons, la distance entre le chopper et le détecteur et ce dernier fournit la position finale du neutron ainsi que son temps d'arrivée.

L'ILL est doté de plusieurs spectromètres à temps de vol, dont la taille caractéristique est de l'ordre de 10 m ; voici le schéma de l'instrument IN6.



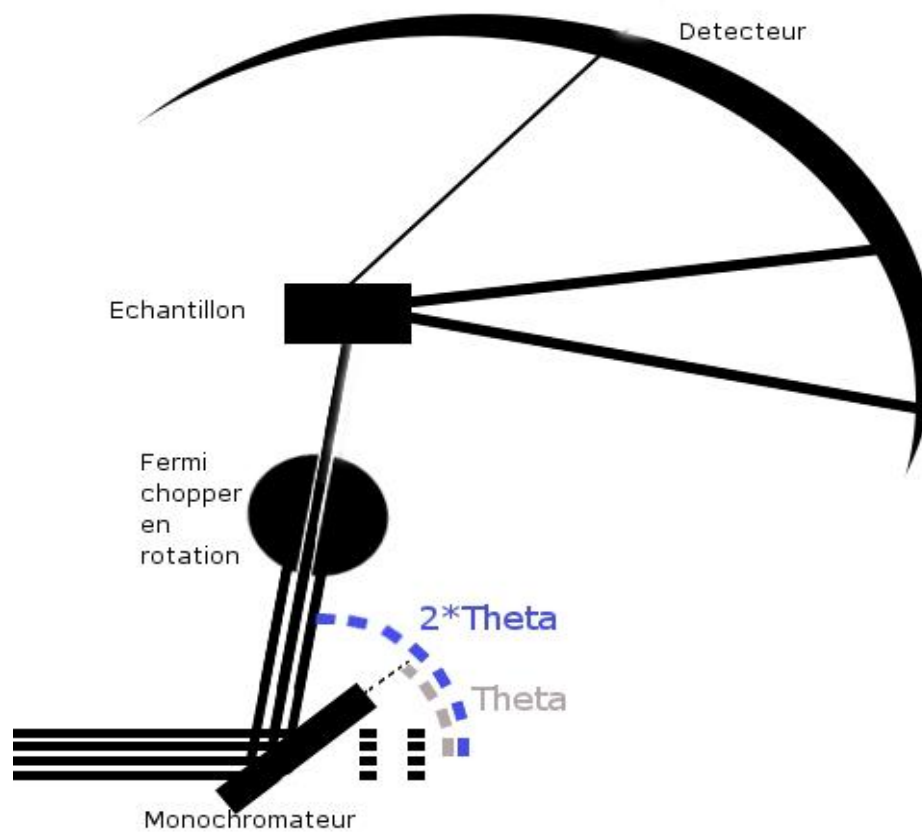
Le guide achemine le flux de neutrons du réacteur vers les instruments. Une partie du flux est diffractée par le monochromateur, selon un angle et une énergie selon la loi de Bragg. S'en suit, une série de *choppers* et collimateurs, qui vont “pulser” le flux et sélectionner une direction incidente vers l'échantillon.

Aspect mécanique:

Le monochromateur, le Fermi chopper et le support de l'échantillon sont les parties mobiles. Ces 3 parties sont mises en rotation autour de la même direction (\vec{z} dans les schémas suivants).

De plus, le flux de neutron est dévié en fonction de la position du monochromateur, ainsi un angle de **Thêta** entre l'orientation du monochromateur et celle du flux de neutron incident va se traduire par un angle **2*Thêta** entre le flux incident et réfléchi. L'alignement des éléments suivant doit donc tenir compte du rayon réfléchi. L'ensemble est mis en rotation autour du monochromateur et va tourner de **2*Thêta**.

Le faisceau de neutrons n'est pas dévié par les autres parties mobiles, sauf dans l'échantillon où il est diffusé et collecté sur un large détecteur.

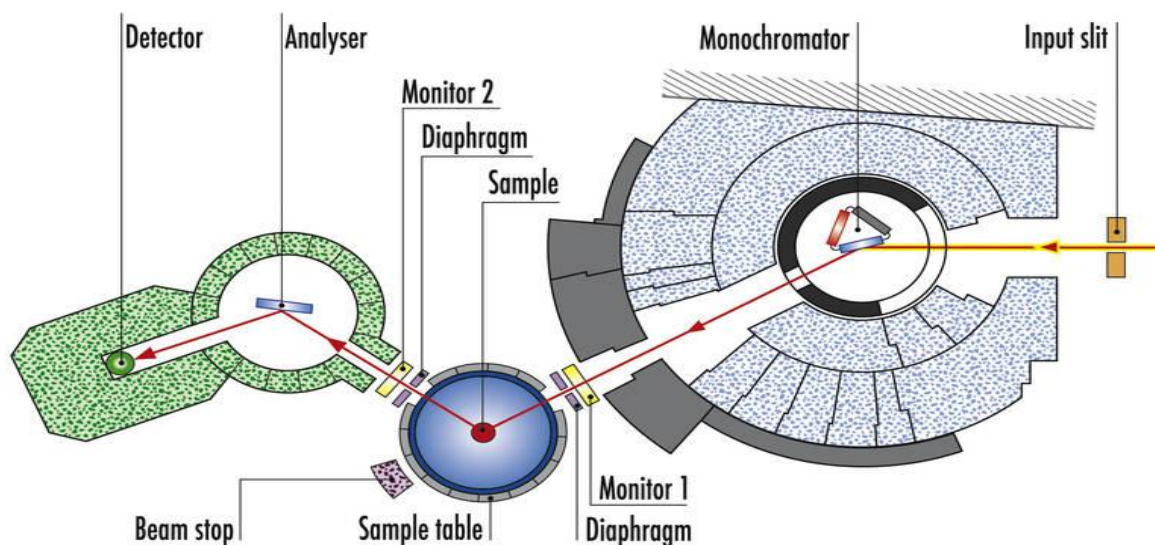


Ci-dessus, le schéma du TOF IN6 mettant en évidence les angles Thêta et 2*Thêta.

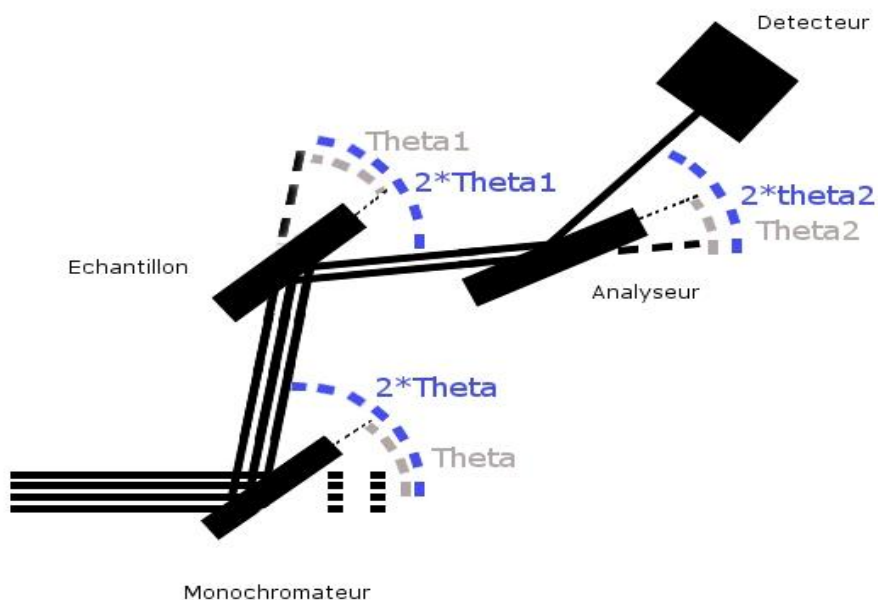
TAS type IN8

Le spectromètre à “trois axes” tire son nom de ses trois parties mobiles, le monochromateur, le support à échantillon et l'analyseur.

L'ILL est doté de plusieurs spectromètres trois axes ; voici le schéma de l'instrument IN8.



Comme pour le TOF IN6, ce TAS utilise un monochromateur pour diriger le flux de neutrons (dont la plage de longueur d'onde est choisie) vers l'échantillon puis l'analyseur qui va sélectionner à nouveau l'énergie des neutrons souhaité pour l'étude. Ci-dessous, le schéma du TAS IN8 mettant en évidence les angles Θ et 2Θ .



Modélisation des spectromètres

Cahier des charges

Nombre de moteurs

Il y a au maximum 3 parties mobiles tournant chacune autour d'un seul axe.

Fonctionnalité

Le modèle doit pouvoir être contrôlé par un opérateur avec ou sans ordinateur (*standalone*).

Le programme du modèle doit pouvoir communiquer avec McStas (définition des angles et autres).

Simplicité

Le montage doit être suffisamment simple et documenté pour pouvoir être construit par quelqu'un n'ayant aucune notion en diffusion neutronique.

Le programme de manipulation doit être le plus simple d'utilisation possible.

Une Solution : Mindstorms NXT

Le Mindstorms NXT est un kit robotique commercialisé par Lego.

Caractéristiques générales

- Une Brique intelligente programmable NXT.
- 3 servo moteurs interactifs.
- Microprocesseur 32 bits ARM7.
- Fonction Bluetooth.
- 1 port USB 2.0.
- 3 ports de sortie pour les moteurs.
- Ecran à cristaux liquides 100x64 pixels.



Langage de programmation

- Langages .NET, tels C sharp ou Visual Basic .NET, grâce à Microsoft Robotics Studio.
- La version Lego de LabVIEW qui permet une prise en main aisée.
- NXC (*Not eXactly C*) qui est un langage proche du C
- RobotC est un autre langage de programmation basé sur C.
- Lejos est une API basé sur le langage Java
- Ubiscript : langage de la plate-forme logicielle Urbi (Universal Real-Time Behavior Interface) .
- Microsoft Robotics Studio
- DialogOS permet de commander les robots par la voix.
- Robolab est une programmation possible du groupe National Instrument.

On remarque dans ces caractéristiques que le NXT peut contrôler simultanément 3 moteurs. Il est connectable à un ordinateur via USB ou Bluetooth. Il est doté d'un écran et de quelques boutons et capteurs de pressions qui permettront de contrôler le modèle également sans ordinateur. Pour finir, il est programmable en différents langage et notamment dans des langages proches du C, ce qui va nous permettre de communiquer avec McStas, lui-même codé en C.

Ce kit Lego semble bien compatible avec nos exigences.

Modélisation mécanique des instruments

La première étape consiste à construire un modèle fonctionnel avant de s'afférer au code.

Description mécanique du TOF IN6

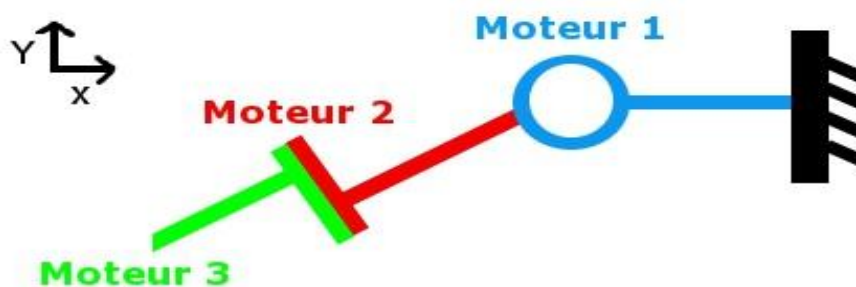


Schéma cinématique du TOF IN6 en 2D

Le moteur 1 entraîne le reste du système (moteur 2 et 3). Les éléments entre le moteur 2 et 3 restent fixes entre eux.

Le moteur 3 entraîne le support à échantillon.

Le moteur 2 entraîne le Fermi Chopper.

Le moteur 1 entraîne également le support du monochromateur, il doit donc assurer la relation d'angles $\Theta/2 * \Theta$. Avec Θ s'appliquant au monochromateur et $2 * \Theta$ s'appliquant au reste du système.

Description mécanique du TAS IN8

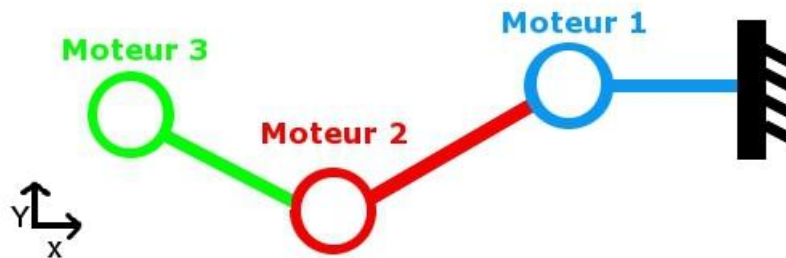


Schéma cinématique du TAS IN8 en 2D

Le moteur 3 entraîne l'Analyseur (en Θ_A) et le détecteur (en $2\cdot\Theta_A$).

Le moteur 2 entraîne le support échantillon (en Θ_E) et tout ce qui est relié au moteur 3 (en $2\cdot\Theta_E$).

Le moteur 1 entraîne le monochromateur (en Θ_M) et tout ce qui est relié au moteur 2 (en $2\cdot\Theta_M$).

Modélisation CAO

La modélisation CAO est importante pour plusieurs raisons. D'une part, le logiciel va nous permettre de créer des instructions de montage étape par étape (extrait visible en **annexe 1**), il va également permettre de commander chez lego les pièces utilisées dans les modèles. Les personnes désireuses de reconstruire cette maquette n'auront qu'à transmettre à Lego la liste des pièces pour recevoir le kit. Les modèles finaux sont visibles en **annexe 2**.

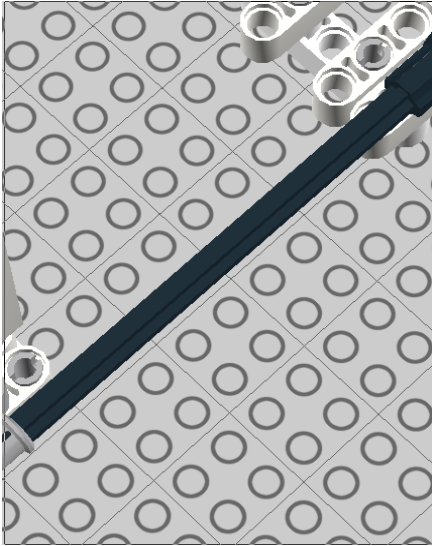


Vue globale des modèles CAO du TAS (à gauche) et du TOF.

Les limites des modèles

Le logiciel de CAO permet de modéliser des mouvements mais il est très limité à ce niveau, puisqu'il ne gère pas les transmissions de mouvements entre engrenages. Les torsions et flexions ne sont pas non plus modélisables. Ainsi, les premiers modèles réalisés se sont révélés en pratique assez inefficaces du fait des torsions et flexions qui s'appliquaient sur les axes de rotations.

Ci-après l'évolution de ces axes de rotation au cours du temps.



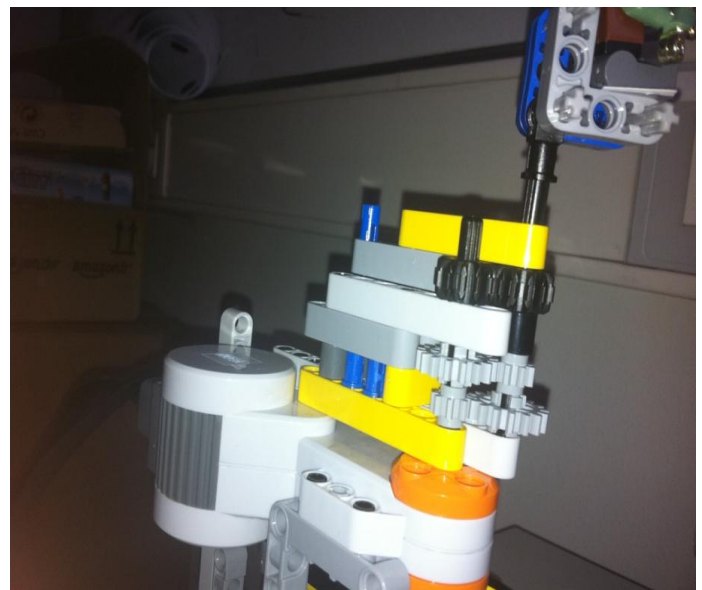
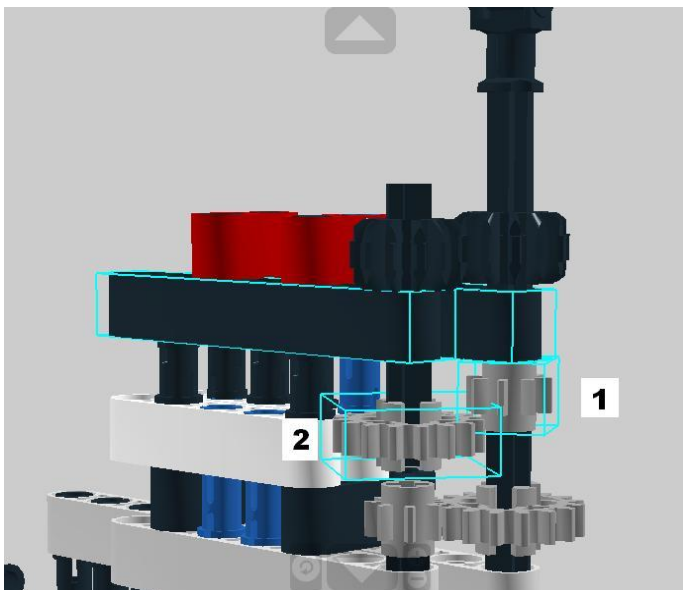
Première version : Poutre simple fixée à la base de l'axe de rotation (loin du moteur).

Seconde version : Poutre double fixée limitant la flexion et fixée à la base de l'axe de rotation.



Dernière version : Poutres doubles pre-contrainte fixées à l'axe par un module solidarisé à l'axe en 2 points, de part et d'autre de l'engrenage.

Une fois les modèles CAO réalisés et validés par le modèle réel, il reste des limites liées au manque de souplesse du logiciel notamment en ce qui concerne le couplage des engrenages. En effet, la détection de collisions entre les engrenages est particulièrement limitante du fait de l'impossibilité d'exercer la moindre torsion afin de les aligner correctement. Concrètement, ce problème m'a obligé à désaxer le premier couple d'engrenage (1 et 2 ci-dessous) du monochromateur. J'ai d'ailleurs dû expliciter ce cas dans la documentation de l'utilisateur pour le sensibiliser à ce défaut de modélisation. L'assemblage réel est lui beaucoup plus tolérant.



Partie supérieure du monochromateur en CAO (à gauche) et en réel.

Code

Le code de micro-programme embarqué dans le NXT est présenté selon 2 versions : La version “Standalone” utilisable sans outils informatique et l'autre version couplée au logiciel de simulation McStas.

La version « standalone »

Cette version a nécessité de penser et créer une IHM reposant sur les interactions avec la brique NXT. A notre disposition, se trouvait les capteurs divers fournis en standard par Lego dans son kit Lego NXT Mindsotrms à savoir un capteur de couleur, un capteur ultrason et 2 capteurs de pressions. Je me suis vite orienté vers les 2 capteurs de pressions qui, tenus dans chaque mains, permettraient de naviguer dans un menu à l'image des joysticks de manettes de jeux vidéos. Le premier capteur sert à mettre en surbrillance un des trois sous-menus (Sélectionneur d'angle du Monochromateur, Sélectionneur de la vitesse du Fermi, et Sélectionneur de l'angle de rotation de l'échantillon pour le TOF et Sélectionneurs des angles des Monochromateurs, Sample et Analyseurs pour le TAS). Le second capteur permet de valider le choix. A l'intérieur de chaque sous-menu, le premier capteur sert à diminuer la valeur à modifier (angle ou vitesse) alors que le second capteur permet de l'augmenter. Le positionnement de la maquette se fait après chaque relâchement des deux capteurs. Appuyer sur les deux capteurs simultanément permet de revenir au menu principal.

Utilisation

a. Lancer le programme

Le menu du programme apparaît à l'écran du NXT. Ce menu va permettre de naviguer entre les 3 sous-menus liés aux 3 moteurs et ainsi de spécifier les angles ou les vitesses de rotation.

Le bouton **G** permet de faire défiler les sous-menus :

Pour le TOF : **Monochromator, Fermi-Chopper** et **Sample**

Pour le TAS : **Monochromator, Sample** et **Analyser**

b. Entrer dans un des sous-menus

Le bouton **D** permet de rentrer dans le sous-menu sélectionné. Pour sortir du sous-menu, il suffit de presser en même temps les boutons **D** et **G**.

ATTENTION : Pour le TAS, il n'est pas possible de rentrer les paramètres du Sample ou de l'Analyser avant celui du monochromateur. Cette mesure vise à réduire les problèmes de précision.

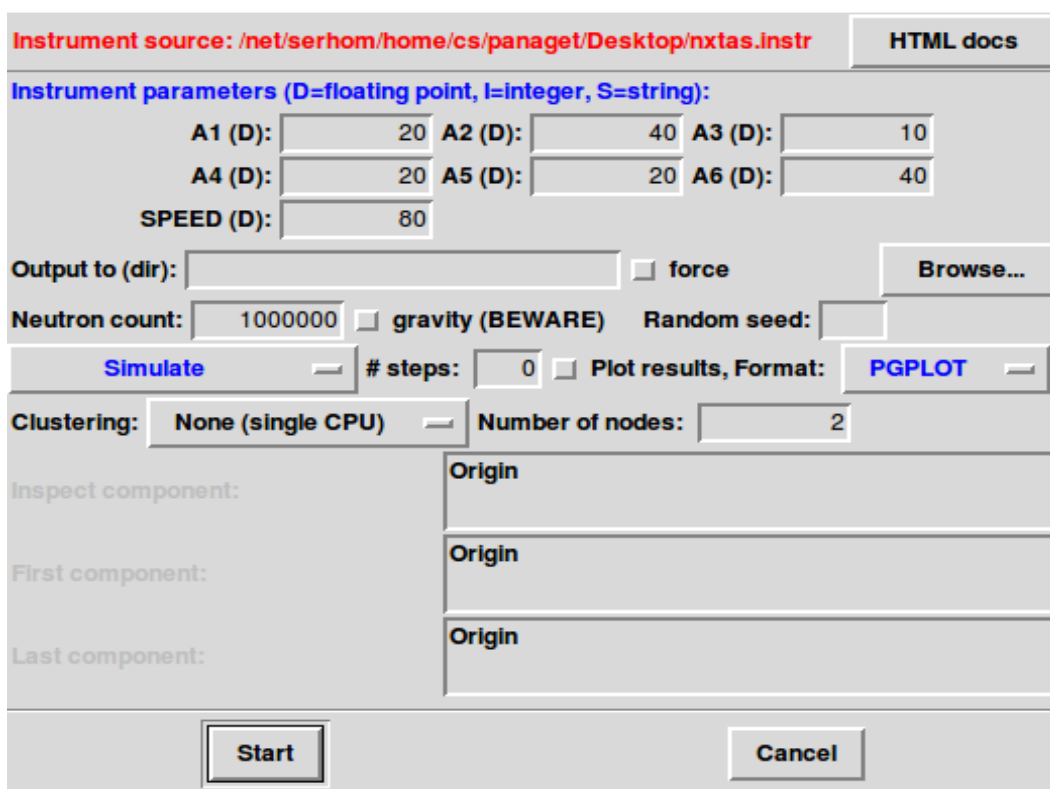
c. Spécifier les paramètres

Dans chaque sous-menu, utiliser **G** ou **D** pour augmenter ou diminuer la valeur de l'angle ou de la vitesse de rotation sélectionnée.

Comportement de la maquette : Une fois la valeur choisie, le moteur se positionne. Chaque moteur est asservi par une boucle de régulation jusqu'à ce que toutes les consignes de positionnement soient satisfaites. Cela permet de corriger tout décalage de positionnement des autres moteurs lors d'un déplacement

La version couplée à McStas

McStas est un logiciel de simulation d'instruments. Chaque instrument fait appel à des « composants » externes, ainsi, un TOF est une succession d'appels de guides, de monochromateurs, d'un Fermi chopper et d'un échantillon et d'un détecteur. Lors du lancement de la simulation de l'instrument, McStas ouvre une boîte de dialogue afin que l'utilisateur spécifie les paramètres des « composants ». Dans le cas du TOF, il faut notamment spécifier les angles de rotations du monochromateur (A1 et A2) et l'angle de rotation du sample (A3).



The image shows the McStas simulation dialog box. At the top, it displays the 'Instrument source' as '/net/serhom/home/cs/panaget/Desktop/nxtas.instr' and a link to 'HTML docs'. Below this, it lists 'Instrument parameters' with fields for A1 (D): 20, A2 (D): 40, A3 (D): 10, A4 (D): 20, A5 (D): 20, A6 (D): 40, and SPEED (D): 80. There is an 'Output to (dir):' field with a 'Browse...' button and a 'force' checkbox. The 'Neutron count' is set to 1000000, with a 'gravity (BEWARE)' checkbox and a 'Random seed' field. A 'Simulate' button is next to the '# steps:' field (set to 0), followed by a 'Plot results, Format:' dropdown set to 'PGPLOT'. The 'Clustering:' dropdown is set to 'None (single CPU)', and the 'Number of nodes:' is set to 2. On the right, there are three 'Inspect component:' labels corresponding to 'Origin' text boxes. At the bottom, there are 'Start' and 'Cancel' buttons.

Boite de dialogue McStas permettant de rentrer les paramètres de la simulation

Dans cette version, McStas va directement diriger les modules de la maquette. Le code NXC permet de charger un fichier texte et de traiter les informations qui y sont contenus. Or, il est facile, avec McStas, de générer un fichier texte contenant les informations de configuration et ensuite de le charger automatiquement dans le NXT. Il suffit de créer un « component » qui ne va pas influencer sur le flux de neutron et qui prend en entrée les paramètres requis, puis qui écrit un fichier texte et l'envoie au NXT à chaque nouvelle configuration (cf **Annexe 3**). Ce système nous permet d'appeler le component dans l'instrument de notre choix sans modifications majeures de celui-ci.

Fichier texte généré et envoyé au NXT :	
<p><i># TAS Parameters for nxTAS v1.0</i></p> <p><i># generated by McStas on 12-12-2012</i></p> <p>A1=45 ;</p> <p>A2=90 ;</p> <p>A3=15 ;</p> <p>A4=30 ;</p> <p>A5=35 ;</p> <p>A6=60 ;</p> <p>SPEED =100 ;</p>	<p><i># TOF Parameters for nxTOF v1.0</i></p> <p><i># generated by McStas on 12-12-2012</i></p> <p>A1=10 ;</p> <p>A2=20 ;</p> <p>A3= 20 ;</p> <p>Fermi speed = 60 ;</p>

Le traitement, par le NXT, des données rentrées par l'utilisateur devra tenir compte des limites du modèle. Ainsi dans la réalité $A(I)$ et $A(I+1)$, avec I un entier supérieur à 0, ne sont pas nécessairement liés par la relation $A(I+1)=2*A(I)$ alors que la limitation du nombre de moteurs à 3 nous impose cette relation. Le code du NXT va alors assurer cette relation en prenant la moyenne arithmétique des deux angles.

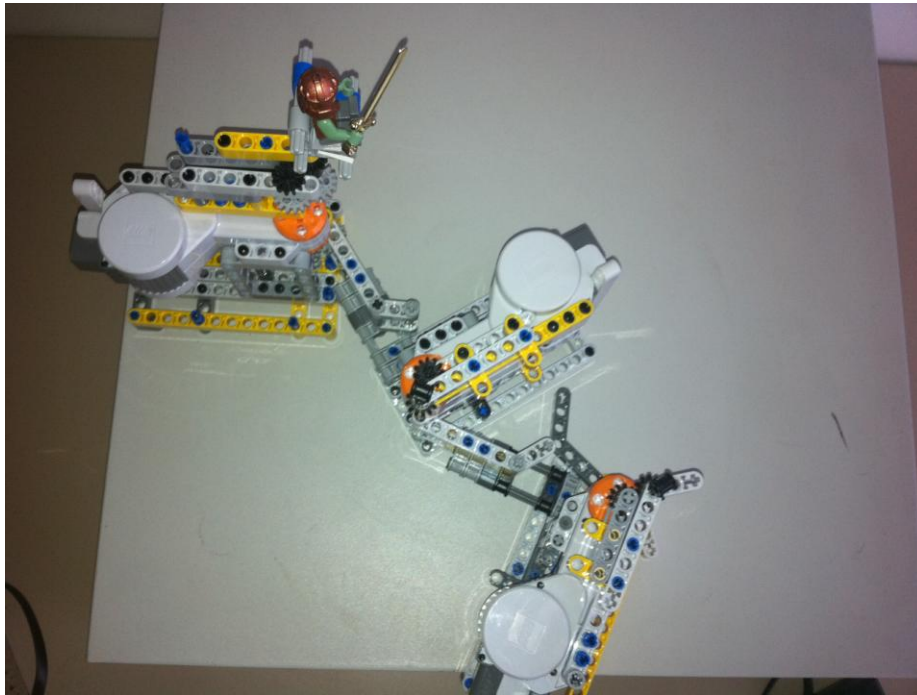
Dans le cas de l'exemple nxTAS ci-avant, $A5_{\text{new}} = (2*A5+A6)/4 = 32.5$ et $A6_{\text{new}} = 2*A5_{\text{new}} = 65$.

Utilisation

b/ Lancer le programme

Le programme embarqué va chercher et charger le fichier texte tout au long de son exécution. Il est possible de charger de nouveaux paramètres à n'importe quel moment. Le temps de transfert est alors de l'ordre de la demi-seconde.

Comportement de la maquette : La maquette se positionne en buté (cf. photo ci-après) à chaque nouveaux jeux de paramètres lus. Une fois le système à l'arrêt, le positionnement demandé débute. Lorsque le positionnement se termine, un signal sonore retentit.



Maquette du TAS en buté

Caractéristiques de la maquette

Une succession de tests m'ont permis d'établir les caractéristiques suivantes.

Dimensions du support (cm) : $W \times D = 45 \times 45$

Dimensions (cm) : $W \times D \times H = 22.60 \times 33.49 \times 17.29$

Temps de réponse (s) : 0.5

Temps de positionnement moyen (s) : 5

Précision moyenne globale (deg) : 5-10

Asservissement des moteurs

Un des problèmes majeur de précision de la maquette, une fois les problèmes de torsion et de flexion résolus, réside dans les phénomènes d'interactions entre les moteurs. Ce problème est spécifique au TAS.

Par exemple : une fois l'échantillon positionné par le moteur support du monochromateur, le positionnement de l'analyseur entraîne un mouvement de l'échantillon qui était correctement positionné et donc qui se retrouve dans une position non souhaitée.

Une solution consiste à asservir les moteurs à leurs positions avant et après chaque positionnement grâce aux moteurs qui sont dotés de codeurs. Néanmoins, le codeur ne retourne pas une valeur absolue mais relative, si bien que les limitations de celui-ci impliquent de faire le zéro avant toute réutilisation des fonctions l'appelant.

Code asservissant le moteur A :

```
if (MotorTachoCount(OUT_A)!=0) {  
    // Si il y a mouvement inopiné du moteur A  
    // rotation du moteur A pour remettre le codeur à 0 et à 90% de la puissance du moteur.  
    RotateMotor(OUT_A,90, - MotorTachoCount(OUT_A));  
    ResetTachoCount(OUT_A); // Remise à 0 du codeur.  
}
```

Pour limiter ces effets d'interactions entre les moteurs, il a été imposé à l'utilisateur de définir dans un premier temps l'angle du Monochromateur, puis l'angle de l'échantillon et pour finir l'angle de l'analyseur. Par ailleurs, les asservissements ont lieu après chaque positionnement d'un moteur et l'ensemble des moteurs est asservi en boucle afin d'éviter les interactions entraînées par les repositionnements (cf **Annexe 4**). Ainsi l'ensemble du système se met en mouvement au même moment et donc tend à revenir à sa position d'équilibre.

Package final

Une fois les problèmes techniques réglés et la maquette validée, j'ai entrepris de réaliser le package final contenant un manuel utilisateur, un guide de construction, ainsi que tous les fichiers nécessaires.

Organisation du Package :

nxTAS-nxTOF_package

Installer NXT ubuntu - transferer les programmes.doc

Modeles CAO - liste pieces.doc

Utiliser les maquettes.doc

CAO

Liste TAS.txt

Liste TOF.txt

TAS building instructions.pdf

TOF building instructions.pdf

TAScommande.lxf

TAS with.lxf

TOFcommande.lxf

TOFwith.lxf

TOFTAScommande.lxf

McStas_components

NXTAS.comp

NXTOF.comp

nxtas.inst

nxtof.inst

IN8_NXT.inst

IN6_NXT.instr

Programs

nxTAS.nxc

nxTOF.nxc

nxTAS_SA.nxc

nxTOF_SA.nxc

L'ensemble des fichiers sont décrits dans les 3 documents *word* du package et disponible en **Annexe 5**.

Conclusion

Ce stage était une occasion pour moi de travailler dans un cadre unique qu'est un réacteur de recherche européen. J'ai pu me familiariser avec des concepts avancés de diffusion neutronique. Par ailleurs, le projet qui m'a été confié me permet d'explorer de nombreux domaines et ce sur toutes les étapes de son élaboration. J'ai alors successivement établi le cahier des charges du projet, réalisé un modèle CAO des modèles TAS et TOF, fait des tests mécanique en traction et flexion, codé une IHM, mis en place l'interaction avec McStas et finalisé le package distribuable. J'ai donc pu explorer plein de facettes de l'ingénierie système, ce stage répond totalement à mes attentes.

Annexes

Index thématique

A	mécanique.....	
Asservissement.....	N.....	40
C	NXT.....	
CAO	S.....	29, 94
Code.....	standalone.....	34
Coupe.....	T.....	99
I	TAS.....	
IN6.....	TOF.....	16
IN8.....	U.....	23
M	Ubuntu	
McStas.....		36

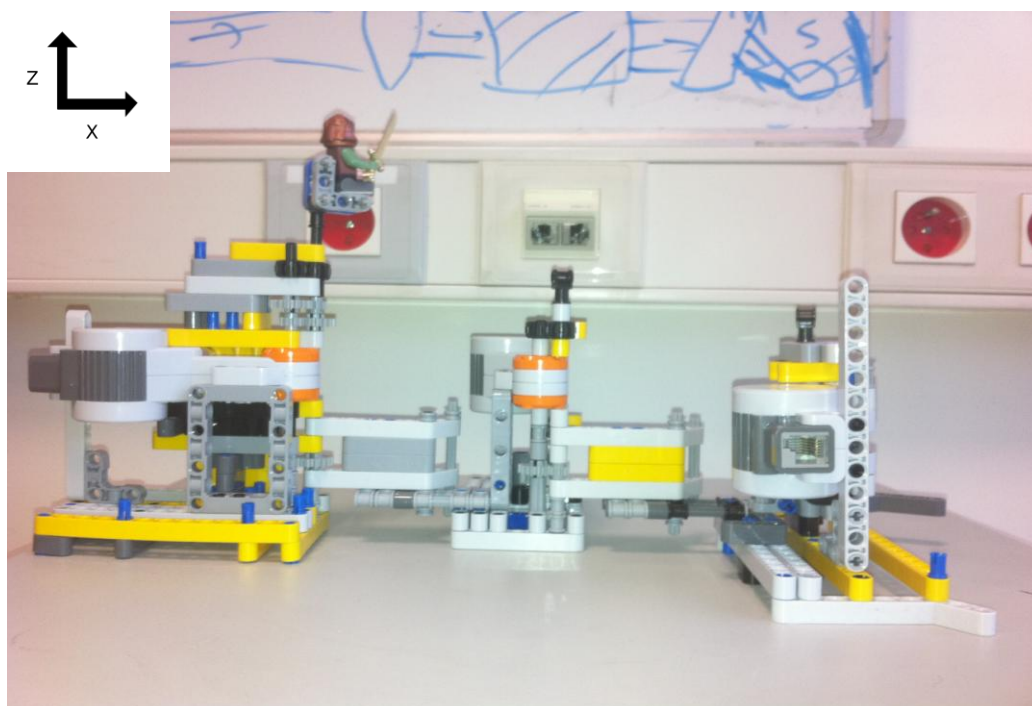
Annexe 1 : Instructions de montage

Exemple d'instructions du TAS

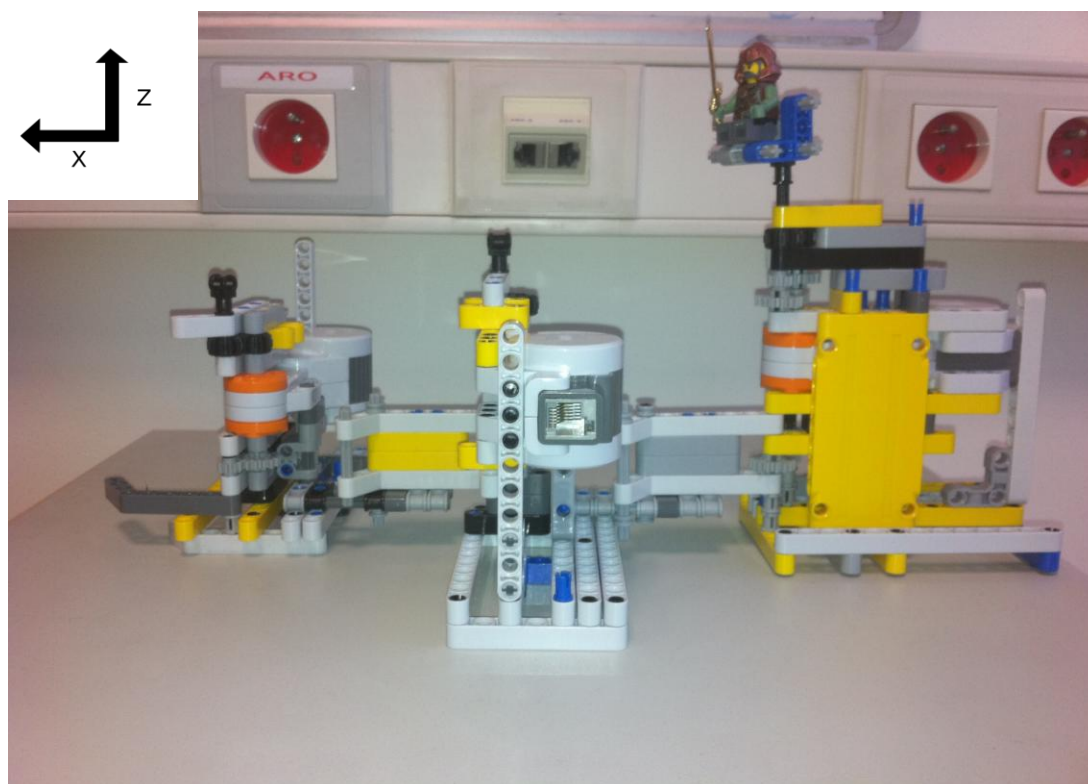
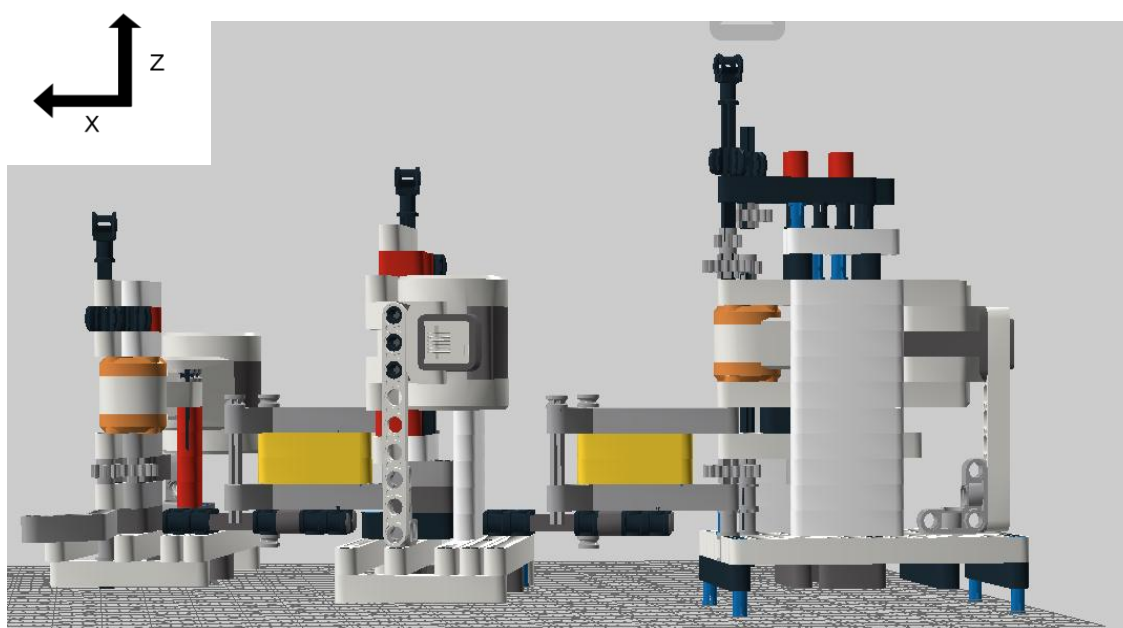


Il y a environ 100 pages par maquette.

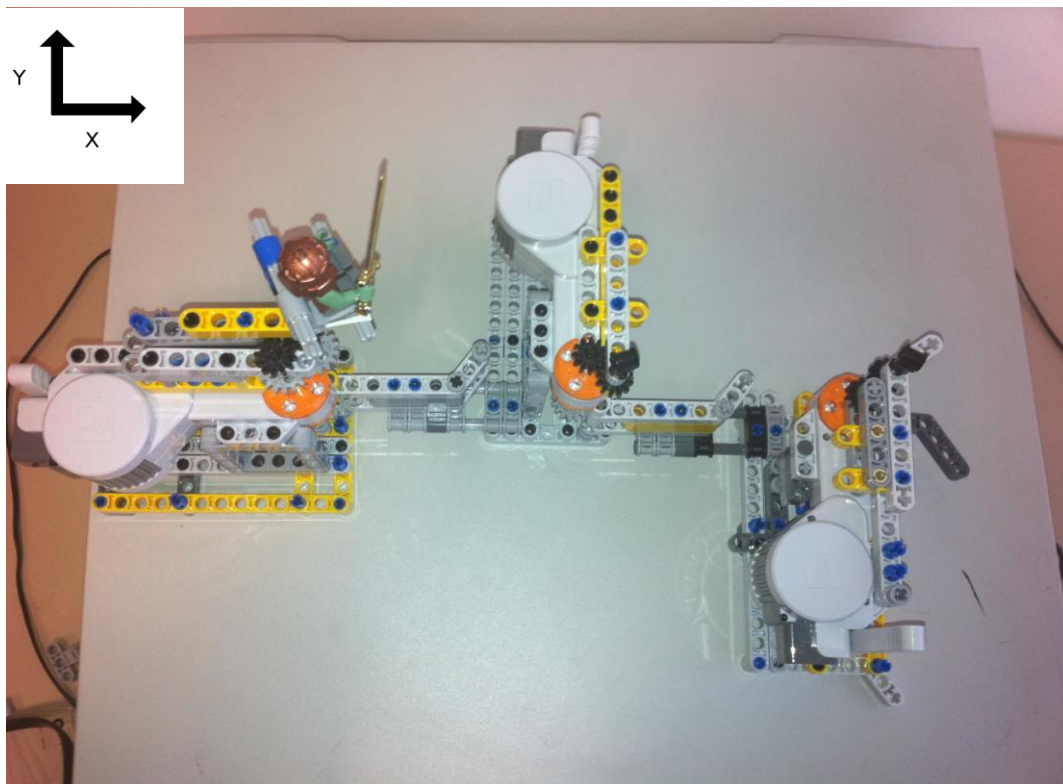
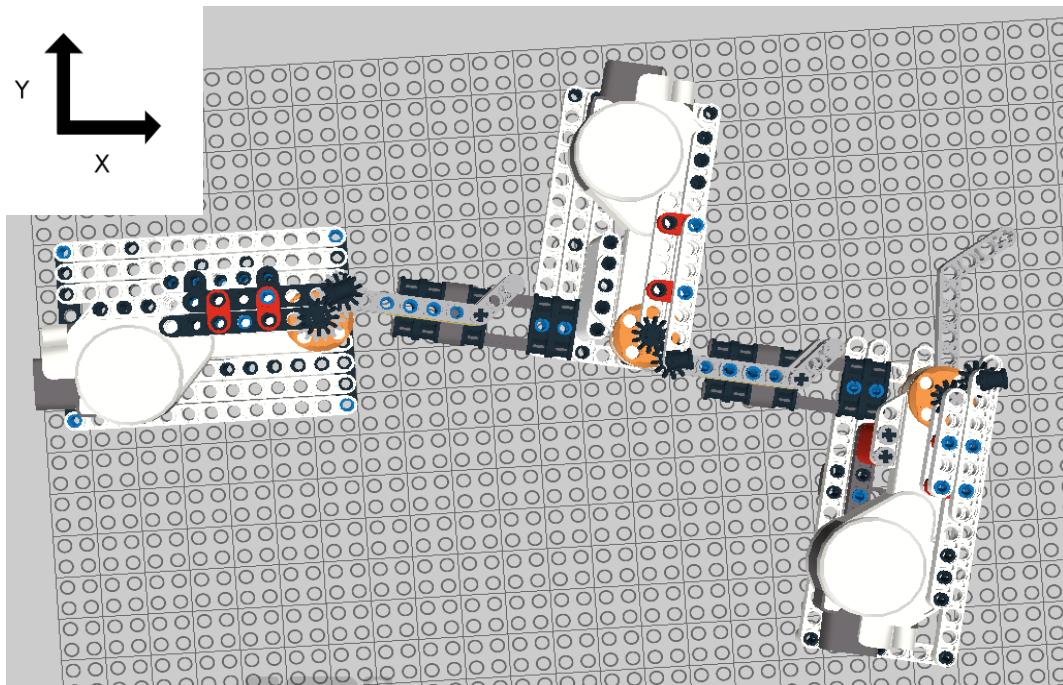
Annexe 2 : Coupe du modèle TAS CAO/réal et TOF CAO.



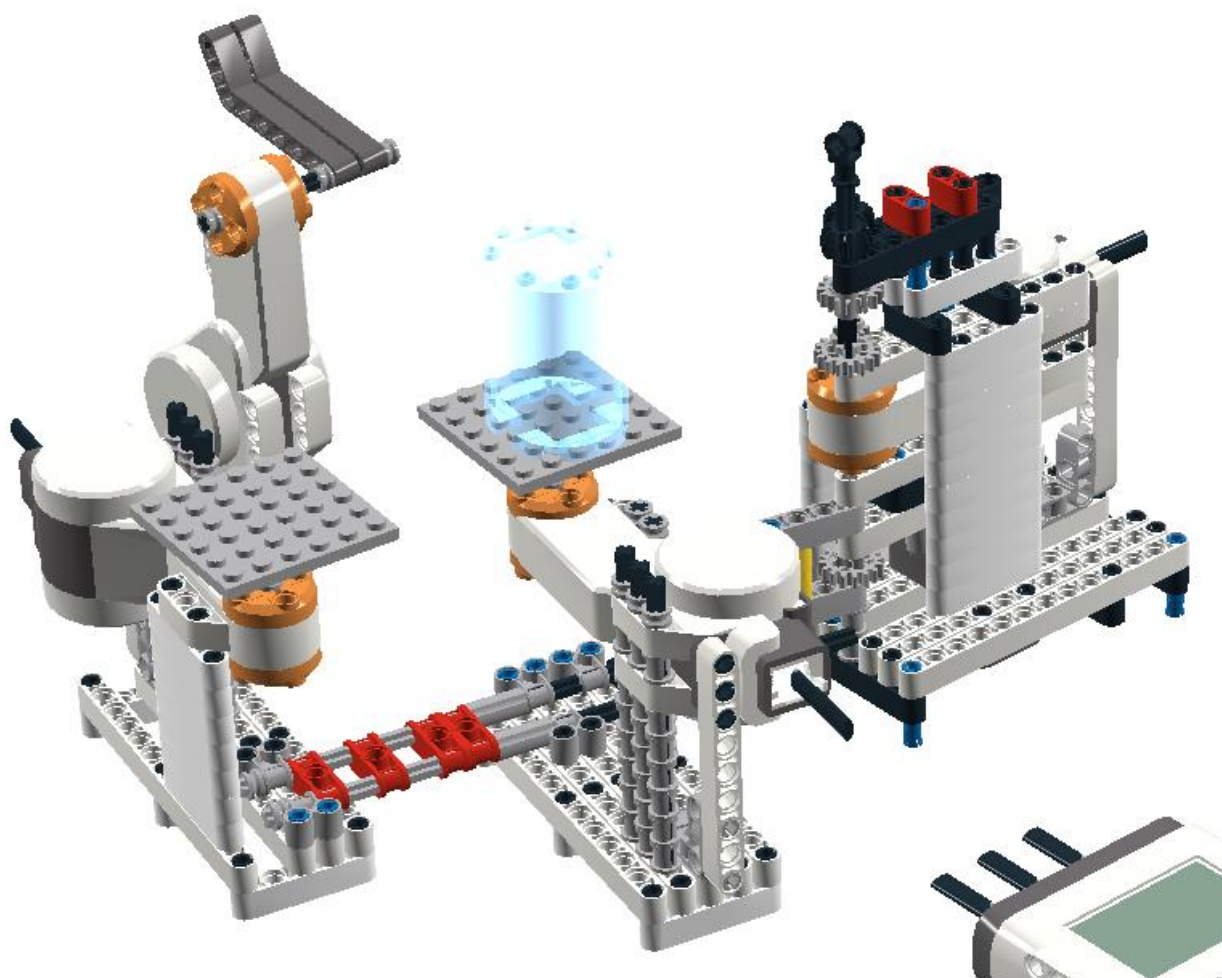
Vue transversale droite TAS



Vue transversale gauche TAS



Vue de dessus TAS



Vue globale du TOF

Annexe 3 : Component et instruments nxTAS et nxTOF.

NXTAS.comp

```

DEFINE COMPONENT NXTAS
SETTING PARAMETERS (A1=0, A2=0, A3=0, A4=0, A5=0, A6=0, SPEED=80)
STATE PARAMETERS (x, y, z, vx, vy, vz, t, s1, s2, p)

INITIALIZE
% {
    FILE *f = fopen("nxtas.txt", "w");
    fprintf(f, "#TAS Parameters for nxTAS v1¥n");
    fprintf(f, "#generated by McStas on 12/12/2012¥n");
    fprintf(f, "¥n");
    fprintf(f, "A1=%g;¥n", A1);
    fprintf(f, "A2=%g;¥n", A2);
    fprintf(f, "A3=%g;¥n", A3);
    fprintf(f, "A4=%g;¥n", A4);
    fprintf(f, "A5=%g;¥n", A5);
    fprintf(f, "A6=%g;¥n", A6);
    fprintf(f, "¥n");
    fprintf(f, "SPEED=%g;¥n", SPEED);
    fclose(f);
    //send the file *.txt to the NXT
    system("echo y | t2n -put nxtas.txt");
% }
END

```

NXTOF.comp

```

DEFINE COMPONENT NXTOF
SETTING PARAMETERS (A1=0, A2=0, A3=0, FSPEED=0)
STATE PARAMETERS (x, y, z, vx, vy, vz, t, s1, s2, p)
INITIALIZE
%{
    FILE *f = fopen("nxtof.txt", "w");
    fprintf(f, "#TOF Parameters for nxTOF v1¥n");
    fprintf(f, "#generated by McStas on 12/12/2012¥n");
    fprintf(f, "¥n");
    fprintf(f, "A1=%g;¥n", A1); //A1 & A2 : monochromator angle
    fprintf(f, "A2=%g;¥n", A2);
    fprintf(f, "A3=%g;¥n", A3); //A3 & A4 : Sample angle
    fprintf(f, "Fermi speed=%g¥n", FSPEED);
    fclose(f);
    //send the file *.txt to the NXT
    system("echo y | t2n -put nxtof.txt");
}%}
END

```

nxtas.instr

```

DEFINE INSTRUMENT nxtas(
    A1=20, A2=40, A3=10, A4=20, A5=20, A6=40, SPEED=80)
TRACE
COMPONENT Origin = Progress_bar()
    AT (0, 0, 0) ABSOLUTE
COMPONENT nxt=NXTAS(
    A1=A1, A2=A2, A3=A3, A4=A4, A5=A5, A6=A6, SPEED=SPEED)
    AT (0, 0, 0) ABSOLUTE
END

```

nxtof.instr


```
DEFINE INSTRUMENT nxtof(  
    A1=20, A2=40, A3=40, FSPEED=50)  
  
TRACE  
  
COMPONENT Origin = Progress_bar()  
    AT (0, 0, 0) ABSOLUTE  
  
COMPONENT nxt=NXTOF (A1=A1, A2=A2, A3=A3, FSPEED=FSPEED)  
    AT (0, 0, 0) ABSOLUTE  
  
END
```

ILL_H15_NXT_IN6.instr

```

/*****
*****
*****

```

```

*
* McStas, neutron ray-tracing package

```

```

* Copyright (C) 1997-2010, All
rights reserved

```

```

* Risoe National Laboratory,
Roskilde, Denmark

```

```

* Institut Laue Langevin, Grenoble,
France

```

```

*
* Instrument: Light_H15_NXT_IN6

```

```

*
* %Identification

```

```

* Written by: <a
href="mailto:farhi@ill.fr">Emmanuel
Farhi</a>

```

```

* Date: 17th Jan 2005.

```

```

* Origin: <a
href="http://www.ill.fr">ILL
(France)</a>

```

```

* Release: McStas 1.12b

```

```

* Version: 0.1

```

```

* %INSTRUMENT_SITE: ILL

```

```

*
* The IN6 Time-of-Flight simulation,
positioned as the first instrument in the

```

```

* cold guide H15 (Nickel coating) at the
ILL.

```

```

*

```

```

* %Description

```

```

*

```

```

* IN6 is a time focussing time-of-flight
spectrometer designed for quasielastic
and

```

```

* inelastic scattering for incident
wavelengths in the range of 4 to 6 Angs.

```

```

*

```

```

* An intense beam is extracted from the

```

H15 guide by a vertically focussing

* monochromator array. It consists of three composite pyrolytic graphite

* monochromators using the full height (20 cm) of the guide and focussing the beam

* at the sample position. In order to minimise the interference with the

* subsequent instruments, the monochromator can deliver only four wavelengths:

* 4.1; 4.6; 5.1; and 5.9 Angs. The second order reflection from the graphite

* monochromator is removed by a beryllium-filter cooled at liquid nitrogen

* temperature.

* To achieve the time-focussing condition, the beam is pulsed by a Fermi chopper.

* It has a small slot length to ensure a good transmission. The normal distance

* between the Fermi chopper and the sample is 38 cm. To prevent frame-overlap when

* the chopper is rotating faster than 7500 rpm, a suppressor chopper is placed

* before the Fermi chopper and rotates in phase with the latter.

*

* The secondary spectrometer consists first of an evacuated sample area. The

* detector bank is entirely covered with detector boxes, thus avoiding the

* inconvenience of moving the counters.

*

* This instrument model contains the complete H15 guide, a triple monochromator

* (using the GROUP), two Fermi Choppers (including one background chopper), a

* liquid sample handling coherent and incoherent processes (elastic and inelastic)

* with multiple scattering, customized monitors, and the SPLIT mechanism to

* improve the statistics.

*

* Example: mcrun
ILL_H15_NXT_IN6.instr -n 1e4 --no-output-files LAMBDA=4.14000

*

* %Parameters

* INPUT PARAMETERS:

* LAMBDA: [Angs] wavelength within 4.14/4.6/5.12/5.92

* DLAMBDA: [Angs] wavelength HALF spread. default is 0.075

* SPEED: [rpm] Fermi chopper speed. -1=auto, 0=stopped in open pos.

* RATIO: [1] Suppressor speed ratio. -1=no suppressor.

* PHASE: [deg] Fermi phase w/r to Suppressor. -360=auto

* M1: monochromator motor 1 position. -1=auto [coder values]

* M2: monochromator motor 2 position. -1=auto [coder values]

* M3: monochromator motor 3 position. -1=auto [coder values]

* MONITOR: monitor preset [something like time in s]

* CHA_WIDTH: [us] channel width. -1=auto

* TOF_CHA_RESOL: [1] number of channels.

* TOF_DELAY: [us] TOF delay. -1=auto

* ELPEAK: [1] elastic peak channel. -1=auto

* m: [1] supermirror guide m-value. 1 for Ni, 1.2 for Ni58, 2-4 for SM

* mFC: [1] supermirror FermiChopper coating m-value

* Sqw_coh: [str] coherent S(q,w) file name

```

* Sqw_inc: [str] incoherent S(q,w)
file name

* radius_o: [m] outer radius of
sample hollow cylinder

* thickness: [m] thickness of sample
hollow cylinder

* A33 : [deg] sample
orientation

*

* %Link

* <a
href="http://www.ill.fr/YellowBook/IN6/
home.html">The IN6@ILL Yellow
Pages</a>

* %Link

* RScherm et al, "Another Time of
Flight Spectrometer", ILL Report
76S235, 1976

* %Link

* RScherm, "A high-resolution
spectrometer ...", report Jul -295-NP,
Kernforschungsanlage Julich, 1965

* %Link

* Y.Blanc, "Le spectrometre a temps de
vol IN6", ILL Report 83BL21G, 1983

* %Link

* K.H.Beckurts et al, Neutron physics,
Kernforschungsanlage Karlsruhe, 1964
(p317)

* %Link

* RScherm and TSpringer, "Proposal of
a multiple Chopper...",
Kernforschungsanlage Julich, 19xx

*

* %End

*****
*****
*****/

DEFINE INSTRUMENT
ILL_H15_NXT_IN6(LAMBDA=4.14,
DLAMBDA=0.075, SPEED=-1,

M1=-1, M2=-1, M3=-1, MONITOR=1,
CHA_WIDTH=-1,

TOF_DELAY=-1,
TOF_CHA_RESOL=128, ELPEAK=-1,
RATIO=1, m=1, mFC=0, PHASE=-360,

string Sqw_coh="Rb_liq_coh.sqw",
string Sqw_inc="Rb_liq_inc.sqw",

radius=0.01, thickness=0.005, A33=20)

DECLARE

%{

%include "monitor_nd-lib"

/* VCS (H1) source parameters */

double sT1=216.8, sI1=1.24e+13;

double sT2=33.9, sI2=1.02e+13;

double sT3=16.7, sI3=3.0423e+12;

double sLambda=4.14, sDLambda=0.1;

/* H15 guide coating parameters */

double gR0 = 1;

double gQc = 0.021;

double gAlpha = 4.07;

double gW = 1.0/300.0;

double gMvalue = 1;

/* H15 gaps and Al windows parameters
*/

double Al_Thickness = 0.002;

double gGap = 0.001;

/* H15 guide curvatures */

double gRh = 2700; /* upwards */

/* H15 guide section parameters (total
length/number of elements) */

double gH = 0.2;

double L_H15_2 = 5.5 / 6, Rh_H15_2
= 0;

double L_H15_3 = 9.973/10, Rh_H15_3
= 0; /* end: d ~ 15.5 moderator at -5.9 */

double L_H15_4 = 6.973/7, Rh_H15_4
= 0; /* end: d ~ 22.5 ... */

double L_H15_5 = 4.75 / 5, Rh_H15_5
= 0; /* VTE is at the end of this section */

double L_H15_6 = 11.473/12, Rh_H15_6
= 0; /* end of H15 MAN spec sheet after 2
elements */

double L_H15_7 = 9.473/10, Rh_H15_7
= 0; /* end: IN6 */

double L_H15_8 = 5.573/6, Rh_H15_8
= 0; /* end MAN spec drawing (d ~ 55.3):
D7 */

double L_H15_9 = 1.25, Rh_H15_9
= 0;

/* capture flux positions from moderator:
21.4 28.4 61.2 */

/* variables for IN6 */

double DM = 3.355; /* mono d-
spacing (Angs) */

double mos = 40;

double RV = 3;

/* Bragg angles of the 3 monochromators
*/

double A1;

double A2;

double A3;

double LME = 2.1; /* distance
monochromator 2 <--> sample */

double LMM = 0.030; /* distance
between 2 monochromators */

double LED = 2.483; /* distance sample
<--> detector */

double LCE = 0.395; /* distance fermi
chopper <--> sample */

double LCC = 0.2; /* [m] Chopper1-
Chopper2 distance */

double Frequency, vi;

double ref_phas=0, phas_ferm;

double iTOF_DELAY; /* time of
arrival at sample position, from source */

double iCHA_WIDTH;

```

```

double iTOF_CHA_RESOL;

double iELPEAK;

double iRATIO;

double iSPEED;

double A2cradle;

double iPHASE, period;

/* monitoring monochromator index for
neutrons being sent to IN6 */

MONND_DECLARE(MonokMonitor);

/* flags per event type */

char flag_ci, flag_co, flag_ct; /* cryo-
in/out container */

char flag_single, flag_multi;

/* monitoring sample env */

double ki_x, ki_y, ki_z;

double kf_x, kf_y, kf_z;

double dq=0, dw=0, vf;

char opt1[256]; /* options for
Monitor_nD */

char opt2[256];

%}

INITIALIZE

%{

    double chopper_const = 252.77;
    /* constant in chopper SPEED formula */

    double Ki, Ei, theta;

    double tmin, tmax;

    double dE = 0.0; /* energy transfer */

    /* VCS/H15: transfert guide parameters
for components */

    if (m)      gMvalue = m;

    if (LAMBDA) sLambda = LAMBDA;

    if (DLAMBDA) sDLambda =
DLAMBDA;

    Ki = 2*PI/sLambda;

    vi = K2V*fabs(Ki);

    Ei = VS2E*vi*vi;

    /* H15: Element rotations = Element
length / Curvature Radius * RAD2DEG
*/

    if (gRh) {

        Rh_H15_2 = L_H15_2
/gRh*RAD2DEG;

        Rh_H15_3 = L_H15_3
/gRh*RAD2DEG;

        Rh_H15_4 = L_H15_4
/gRh*RAD2DEG;

        Rh_H15_5 = L_H15_5
/gRh*RAD2DEG;

    }

    /* IN6: calculate theta angles for 3
monochromators */

    theta = asin(sLambda/DM/2);

    A2 = theta*2;

    A1 =
atan2(LME*sin(A2),(LME*cos(A2)+LMM))
*RAD2DEG;

    A3 =
atan2(LME*sin(A2),(LME*cos(A2)-
LMM))
*RAD2DEG;

    A2 *=RAD2DEG;

    A2cradle = A2;

    RV = 2*LME*sin(theta);

    if (A1<0.0) A1=180+A1;

    if (A2<0.0) A2=180+A2;

    if (A3<0.0) A3=180+A3;

    if (M1 == 0) A1 = 0; else

        if (M1>=0) A1 = -
0.0210199*M1+178.55; else M1 = -(A1-
178.55)/0.0210199;

    if (M2 == 0) A2 = 0; else

        if (M2>=0) A2 = -
0.0210302*M2+182.558; else M2 = -(A2-
182.558)/0.0210302;

    if (M3 == 0) A3 = 0; else

        if (M3>=0) A3 = -
0.0206945*M3+187.566; else M3 = -(A3-
187.566)/0.0206945;

    MONND_USER_TITLE(MonokMonitor,
1, "Monok index");

    /* IN6: compute ToF settings from
LightCustom.Light_Custom_IN6_Calc_
TOF_Choppers */

    {

        double el_t_resol = 0.125; /* [us]
Electronic Time Base */

        ref_phas = 0; /* [deg] Reference
Phase */

        double phase_offset=0; /* [deg]
Phase Offset (added to Fermi phase) */

        double el_delay = 44.875; /* [us]
Default Electronic Delay */

        double speed, chan_width, dead_time,
time_of_flight, trav_time;

        double delta_phase, el_peak_O, delay;

        if (TOF_CHA_RESOL<=0)
iTOF_CHA_RESOL=128; else
iTOF_CHA_RESOL=TOF_CHA_RESOL

```

```

L;

if (RATIO <= 0)    iRATIO = 1;
else iRATIO      =RATIO;

if (ELPEAK >= 0 &&
ELPEAK<=iTOF_CHA_RESOL)
iELPEAK = ELPEAK;

else
iELPEAK=ceil(iTOF_CHA_RESOL/2);

speed =
60*K2V/(DM*cos(the ta)*(LCE+(LED*p
ow((1-dE/Ei),-1.5))));

period    =0.5e6 * 60 * iRATIO /speed;

chan_width =
floor(period/el_t_resol/iTOF_CHA_RES
OL)*el_t_resol;

dead_time = period-
(iTOF_CHA_RESOL*chan_width);

time_of_flight =
(LCC+LCE+LED)/vi*1e6;

trav_time = LCC/vi*1e6;

delta_phase = (trav_time/period)*180;

phas_ferm = ref_phas + delta_phase;

if (fmod(iRATIO, 2) == 0) phas_ferm
*= 2;

phas_ferm += phase_offset;

el_peak_O = floor((time_of_flight +
el_delay)/chan_width);

delay = (el_peak_O-iELPEAK) *
chan_width;

if (iELPEAK >= el_peak_O) delay +=
period;

if (delay <= 1) delay = 2;

if (PHASE>-180 && PHASE <360)
iPHASE=PHASE; else iPHASE=-
phas_ferm;

if (CHA_WIDTH <=0)
iCHA_WIDTH=chan_width; else
iCHA_WIDTH=CHA_WIDTH;

if (TOF_DELAY <=0)
iTOF_DELAY=delay; else
iTOF_DELAY=TOF_DELAY;

if (SPEED <0) iSPEED=speed;
else iSPEED=SPEED;

Frequency = iSPEED/60;

printf("Instrument Simulation %s
%s)\n", mcinstrument_name,
mcinstrument_source);

printf(" using computed
monochromator take-off angles: %g %g
%g [deg]\n", A1, A2, A3);

printf("Wavelength      [AA]
%g\n", sLambda);

printf("Neutron velocity    [m/s]
%g\n", vi);

printf("Monochr. Bragg angle  [deg]
%g\n", A2/2);

printf("Incident Energy      [meV]
%g\n", Ei);

printf("Focusing Energy
Transfert[meV] %g\n", dE);

printf("Travel time: Supp./Fermi [us]
%g\n", trav_time);

printf("Travel time: Supp./Det. [us]
%g\n", time_of_flight);

printf("TOF Delay          [us]
%g\n", delay);

printf("TOF Dead Time      [us]
%g\n", dead_time);

printf("TOF Period (1 cycle) [us]
%g\n", period);

printf("TOF Channel width    [us]
%g\n", chan_width);

printf("CHOP Fermi Phase      [deg]
%g\n", iPHASE);

printf("CHOP Suppressor Phase
[deg] %g\n", ref_phas);

printf("CHOP Fermi Speed      [rpm]
%g\n", iSPEED);

printf("CHOP Suppressor Speed
[rpm] %g\n", iSPEED/iRATIO);

printf("Number of time channels
%g\n", iTOF_CHA_RESOL);

printf("Current Elastic Peak Ch.
%g\n", iELPEAK);

printf("Elast. peak ch. for 0-delay
%g\n", el_peak_O);

/* chopper to detector */

tmin = time_of_flight*1e-6 -
(iCHA_WIDTH*iELPEAK-
iTOF_DELAY)*1e-6;
}

/* distance to cover to detector from
chopper: LCE+LED

Center time on 0 at Fermi center

LCE from chopper to sample pos

LED from sample to detector

propagation time t_p
=(LCE+LED)/vi;

falls on ELPEAK channel.

Tmin = t_p-iCHA_WIDTH*1e-
6*iELPEAK

Tmax = Tmin +N_CHAN...

*/

tmax =
tmin+iTOF_CHA_RESOL*iCHA_WID
TH*1e-6;

printf("Time window:
min=%g max=%g delay=%g tof-
width=%g [ms]\n", tmin*1000,
tmax*1000, iTOF_DELAY*1e-3, (tmax-
tmin)*1000);

sprintf(opt2, "kxy limits=[0 5] bins=50,
energy limits=[%g %g] bins=40, banana,
parallel",

(Ei-20 < 0 ? 0 : Ei-20), Ei+20);

sprintf(opt1, "angle limits=[0 180]
bins=180, energy limits=[%g %g]
bins=40, banana, parallel",

(Ei-20 < 0 ? 0 : Ei-20), Ei+20);

```

```

AT (0,0,0.21) RELATIVE VCS

%}

/* ----- TRACE -----
----- */

TRACE

COMPONENT Origin =
Progress_bar(profile="profile")

AT (0,0,0) ABSOLUTE

COMPONENT
nxt=NXTOF(A1=A2/2,A2=A2,A
3=A33,FSPEED=Frequency)

AT (0,0,0) ABSOLUTE

COMPONENT VCS = Source_gen(
h      = 0.22,
w      = 0.14,
dist   = 2.525,
xw     = 0.038,
yh     = 0.2,
Lmin   = sLambda-sDLambda,
Lmax   = sLambda+sDLambda,
T1     = sT1,
I1     = sI1,
T2     = sT2,
I2     = sI2,
T3     = sT3,
I3     = sI3,
verbose = 1)

AT (0, 0, 0) RELATIVE Origin

COMPONENT Al_window1 =
Al_window(win_thick=Al_Thickness)

AT (0,0,0.21) RELATIVE VCS

COMPONENT Al_window2 =
Al_window(win_thick=Al_Thickness)

AT (0,0,0.61) RELATIVE VCS

COMPONENT Al_window3 =
Al_window(win_thick=Al_Thickness)

AT (0,0,0.78) RELATIVE VCS

COMPONENT Al_window4 =
Al_window(win_thick=Al_Thickness)

AT (0,0,0.92) RELATIVE VCS

COMPONENT Al_window5 =
Al_window(win_thick=Al_Thickness)

AT (0,0,2.43) RELATIVE VCS

/* H15-1: L=3.17 m in 1 element. no
curvature */

COMPONENT PinkCarter =
Guide_gravity(
w1=0.038, h1=0.2, w2=0.032, h2=0.2,
l=3.170,
R0=gR0, Qc=gQc, alpha=gAlpha,
m=gMvalue, W=gW)

AT (0,0,2.525) RELATIVE VCS

COMPONENT FirstObturator =
Guide_gravity(
w1=0.031, h1=0.2, w2=0.031, h2=0.2,
l=0.228,
R0=gR0, Qc=gQc, alpha=gAlpha,
m=gMvalue, W=gW)

AT (0,0,3.17+0.02) RELATIVE
PinkCarter

/* ***** swimming pool
guide ***** */

/* H15-2: L=5.5 m in 6 elements R
horiz=2700 m */

COMPONENT H15_2 = Arm()

AT (0,0,3.59) RELATIVE PinkCarter

/*

COMPONENT Mon_2_xy =
Monitor_nD(
xwidth=0.03, yheight=0.2,
options='x y, parallel, per cm2, slit')

AT (0,0,0) RELATIVE PREVIOUS

COMPONENT Mon_2_dxdy =
Monitor_nD(
xwidth=0.03, yheight=0.2,
options='dx dy, all auto, parallel, per
cm2, slit')

AT (0,0,0) RELATIVE PREVIOUS

COMPONENT Mon_2_Phic =
Monitor_nD(
xwidth=0.03, yheight=0.2,
options='x y dx dy, all auto, parallel,
per cm2, capture, slit')

AT (0,0,0) RELATIVE PREVIOUS

COMPONENT Mon_2_L = Monitor_nD(
xwidth=0.03, yheight=0.2,
options='lambda, limits=[1 21] bins=20,
parallel, per cm2, slit')

AT (0,0,0) RELATIVE PREVIOUS

/*

COMPONENT H15_2_In =
Al_window(win_thick=Al_Thickness)

AT (0,0,0) RELATIVE PREVIOUS

```

```

COMPONENT H15_2_1 =
Guide_gravity(

    w1=0.03,h1=0.2, w2=0.03, h2=0.2,
    l=L_H15_2,

    R0=gR0, Qc=gQc, alpha=gAlpha,
    m=gMvalue, W=gW)

AT (0,0,Al_Thickness+gGap) RELATIVE
PREVIOUS ROTATED (0,Rh_H15_2,0)
RELATIVE PREVIOUS

COMPONENT H15_2_2 =
Guide_gravity(

    w1=0.03,h1=0.2, w2=0.03, h2=0.2,
    l=L_H15_2,

    R0=gR0, Qc=gQc, alpha=gAlpha,
    m=gMvalue, W=gW)

AT (0,0,L_H15_2+gGap) RELATIVE
PREVIOUS ROTATED (0,Rh_H15_2,0)
RELATIVE PREVIOUS

COMPONENT H15_2_3 =
Guide_gravity(

    w1=0.03,h1=0.2, w2=0.03, h2=0.2,
    l=L_H15_2,

    R0=gR0, Qc=gQc, alpha=gAlpha,
    m=gMvalue, W=gW)

AT (0,0,L_H15_2+gGap) RELATIVE
PREVIOUS ROTATED (0,Rh_H15_2,0)
RELATIVE PREVIOUS

COMPONENT H15_2_4 =
Guide_gravity(

    w1=0.03,h1=0.2, w2=0.03, h2=0.2,
    l=L_H15_2,

    R0=gR0, Qc=gQc, alpha=gAlpha,
    m=gMvalue, W=gW)

AT (0,0,L_H15_2+gGap) RELATIVE
PREVIOUS ROTATED (0,Rh_H15_2,0)
RELATIVE PREVIOUS

COMPONENT H15_2_5 =
Guide_gravity(

    w1=0.03,h1=0.2, w2=0.03, h2=0.2,

l=L_H15_2,

    R0=gR0, Qc=gQc, alpha=gAlpha,
    m=gMvalue, W=gW)

AT (0,0,L_H15_2+gGap) RELATIVE
PREVIOUS ROTATED (0,Rh_H15_2,0)
RELATIVE PREVIOUS

COMPONENT H15_2_6 =
Guide_gravity(

    w1=0.03,h1=0.2, w2=0.03, h2=0.2,
    l=L_H15_2,

    R0=gR0, Qc=gQc, alpha=gAlpha,
    m=gMvalue, W=gW)

AT (0,0,L_H15_2+gGap) RELATIVE
PREVIOUS ROTATED (0,Rh_H15_2,0)
RELATIVE PREVIOUS

COMPONENT H15_2_Out =
Al_window(win_thick=Al_Thickness)

AT (0,0,L_H15_2+gGap) RELATIVE
PREVIOUS

/* gap 0.198 m (VS) */

/* H15-3: L=9.973 m in 10 elements
Rh=2700 m. */

COMPONENT H15_3 = Arm()

AT (0,0,0.198) RELATIVE H15_2_Out

/*

COMPONENT Mon_3_xy =
Monitor_nD(

    xwidth=0.03, yheight=0.2,

    options='x y, parallel, per cm2, slit')

AT (0,0,0) RELATIVE PREVIOUS

COMPONENT Mon_3_dxdy =
Monitor_nD(

    xwidth=0.6, yheight=0.2,

    options='dx dy, all auto, parallel, per
cm2, slit')

AT (0,0,0) RELATIVE PREVIOUS

COMPONENT Mon_3_Phic =
Monitor_nD(

    xwidth=0.03, yheight=0.2,

    options='x y dx dy, all auto, parallel,
per cm2, capture, slit')

AT (0,0,0) RELATIVE PREVIOUS

COMPONENT Mon_3_L = Monitor_nD(

    xwidth=0.03, yheight=0.2,

    options='lambda, limits=[1 21] bins=20,
parallel, per cm2, slit')

AT (0,0,0) RELATIVE PREVIOUS

*/

COMPONENT H15_3_In =
Al_window(win_thick=Al_Thickness)

AT (0,0,0) RELATIVE PREVIOUS

COMPONENT H15_3_1 =
Guide_gravity(

    w1=0.03,h1=0.2, w2=0.03, h2=0.2,
    l=L_H15_3,

    R0=gR0, Qc=gQc, alpha=gAlpha,
    m=gMvalue, W=gW)

AT (0,0,Al_Thickness+gGap)
RELATIVE PREVIOUS ROTATED
(0,Rh_H15_3,0) RELATIVE PREVIOUS

COMPONENT H15_3_2 =
Guide_gravity(

    w1=0.03,h1=0.2, w2=0.03, h2=0.2,
    l=L_H15_3,

    R0=gR0, Qc=gQc, alpha=gAlpha,
    m=gMvalue, W=gW)

AT (0,0,L_H15_3+gGap) RELATIVE
PREVIOUS ROTATED (0,Rh_H15_3,0)
RELATIVE PREVIOUS

```



```

COMPONENT H15_3_3 =
Guide_gravity(
    w1=0.03, h1=0.2, w2=0.03, h2=0.2,
    l=L_H15_3,

    R0=gR0, Qc=gQc, alpha=gAlpha,
    m=gMvalue, W=gW)

AT (0,0,L_H15_3+gGap) RELATIVE
PREVIOUS ROTATED (0,Rh_H15_3,0)
RELATIVE PREVIOUS

COMPONENT H15_3_4 =
Guide_gravity(
    w1=0.03, h1=0.2, w2=0.03, h2=0.2,
    l=L_H15_3,

    R0=gR0, Qc=gQc, alpha=gAlpha,
    m=gMvalue, W=gW)

AT (0,0,L_H15_3+gGap) RELATIVE
PREVIOUS ROTATED (0,Rh_H15_3,0)
RELATIVE PREVIOUS

COMPONENT H15_3_5 =
Guide_gravity(
    w1=0.03, h1=0.2, w2=0.03, h2=0.2,
    l=L_H15_3,

    R0=gR0, Qc=gQc, alpha=gAlpha,
    m=gMvalue, W=gW)

AT (0,0,L_H15_3+gGap) RELATIVE
PREVIOUS ROTATED (0,Rh_H15_3,0)
RELATIVE PREVIOUS

COMPONENT H15_3_6 =
Guide_gravity(
    w1=0.03, h1=0.2, w2=0.03, h2=0.2,
    l=L_H15_3,

    R0=gR0, Qc=gQc, alpha=gAlpha,
    m=gMvalue, W=gW)

AT (0,0,L_H15_3+gGap) RELATIVE
PREVIOUS ROTATED (0,Rh_H15_3,0)
RELATIVE PREVIOUS

COMPONENT H15_3_7 =
Guide_gravity(
    w1=0.03, h1=0.2, w2=0.03, h2=0.2,
    l=L_H15_3,

    R0=gR0, Qc=gQc, alpha=gAlpha,
    m=gMvalue, W=gW)

AT (0,0,L_H15_3+gGap) RELATIVE
PREVIOUS ROTATED (0,Rh_H15_3,0)
RELATIVE PREVIOUS

COMPONENT H15_3_8 =
Guide_gravity(
    w1=0.03, h1=0.2, w2=0.03, h2=0.2,
    l=L_H15_3,

    R0=gR0, Qc=gQc, alpha=gAlpha,
    m=gMvalue, W=gW)

AT (0,0,L_H15_3+gGap) RELATIVE
PREVIOUS ROTATED (0,Rh_H15_3,0)
RELATIVE PREVIOUS

COMPONENT H15_3_9 =
Guide_gravity(
    w1=0.03, h1=0.2, w2=0.03, h2=0.2,
    l=L_H15_3,

    R0=gR0, Qc=gQc, alpha=gAlpha,
    m=gMvalue, W=gW)

AT (0,0,L_H15_3+gGap) RELATIVE
PREVIOUS ROTATED (0,Rh_H15_3,0)
RELATIVE PREVIOUS

COMPONENT H15_3_10 =
Guide_gravity(
    w1=0.03, h1=0.2, w2=0.03, h2=0.2,
    l=L_H15_3,

    R0=gR0, Qc=gQc, alpha=gAlpha,
    m=gMvalue, W=gW)

AT (0,0,L_H15_3+gGap) RELATIVE
PREVIOUS ROTATED (0,Rh_H15_3,0)
RELATIVE PREVIOUS

COMPONENT H15_3_Out =
Al_window(win_thick=Al_Thickness)

AT (0,0,L_H15_3+gGap) RELATIVE
PREVIOUS

/* gap 0.03 m */

/* H15-4: L=6.973 m in 7 elements
Rh=2700 m. Here d_c ~ 21.4 */

COMPONENT H15_4 = Arm()

AT (0,0,0.03) RELATIVE H15_3_Out

/*

COMPONENT Mon_4_xy =
Monitor_nD(
    xwidth=0.03, yheight=0.2,
    options='x y, parallel, per cm2, slit')

AT (0,0,0) RELATIVE PREVIOUS

COMPONENT Mon_4_dxdy =
Monitor_nD(
    xwidth=0.03, yheight=0.2,
    options='dx dy, all auto, parallel, per
cm2, slit')

AT (0,0,0) RELATIVE PREVIOUS

COMPONENT Mon_4_Phic =
Monitor_nD(
    xwidth=0.03, yheight=0.2,
    options='x y dx dy, all auto, parallel,
per cm2, capture, slit')

AT (0,0,0) RELATIVE PREVIOUS

COMPONENT Mon_4_L = Monitor_nD(
    xwidth=0.03, yheight=0.2,
    options='lambda, limits=[1 21] bins=20,
parallel, per cm2, slit')

AT (0,0,0) RELATIVE PREVIOUS

*/

COMPONENT H15_4_In =
Al_window(win_thick=Al_Thickness)

AT (0,0,0) RELATIVE PREVIOUS

```



```

COMPONENT H15_4_1 =
Guide_gravity(
    w1=0.03,h1=0.2, w2=0.03, h2=0.2,
    l=L_H15_4,

    R0=gR0, Qc=gQc, alpha=gAlpha,
    m=gMvalue, W=gW)

AT (0,0,L_H15_4+gGap) RELATIVE
PREVIOUS ROTATED (0,Rh_H15_4,0)
RELATIVE PREVIOUS

COMPONENT H15_4_2 =
Guide_gravity(
    w1=0.03,h1=0.2, w2=0.03, h2=0.2,
    l=L_H15_4,

    R0=gR0, Qc=gQc, alpha=gAlpha,
    m=gMvalue, W=gW)

AT (0,0,L_H15_4+gGap) RELATIVE
PREVIOUS ROTATED (0,Rh_H15_4,0)
RELATIVE PREVIOUS

COMPONENT H15_4_3 =
Guide_gravity(
    w1=0.03,h1=0.2, w2=0.03, h2=0.2,
    l=L_H15_4,

    R0=gR0, Qc=gQc, alpha=gAlpha,
    m=gMvalue, W=gW)

AT (0,0,L_H15_4+gGap) RELATIVE
PREVIOUS ROTATED (0,Rh_H15_4,0)
RELATIVE PREVIOUS

COMPONENT H15_4_4 =
Guide_gravity(
    w1=0.03,h1=0.2, w2=0.03, h2=0.2,
    l=L_H15_4,

    R0=gR0, Qc=gQc, alpha=gAlpha,
    m=gMvalue, W=gW)

AT (0,0,L_H15_4+gGap) RELATIVE
PREVIOUS ROTATED (0,Rh_H15_4,0)
RELATIVE PREVIOUS

COMPONENT H15_4_5 =
Guide_gravity(
    w1=0.03,h1=0.2, w2=0.03, h2=0.2,
    l=L_H15_4,

    R0=gR0, Qc=gQc, alpha=gAlpha,
    m=gMvalue, W=gW)

AT (0,0,L_H15_4+gGap) RELATIVE
PREVIOUS ROTATED (0,Rh_H15_4,0)
RELATIVE PREVIOUS

COMPONENT Mon_5_dxdy =
Monitor_nD(
    xwidth=0.03, yheight=0.2,

    options='x y, parallel, per cm2, slit')

AT (0,0,0) RELATIVE PREVIOUS

COMPONENT Mon_5_Phic =
Monitor_nD(
    xwidth=0.03, yheight=0.2,

    options='x y dx dy, all auto, parallel,
    per cm2, capture, slit')

AT (0,0,0) RELATIVE PREVIOUS

COMPONENT Mon_5_L = Monitor_nD(
    xwidth=0.03, yheight=0.2,

    options='lambda, limits=[1 21] bins=20,
    parallel, per cm2, slit')

AT (0,0,0) RELATIVE PREVIOUS

*/

COMPONENT H15_5_In =
Al_window(win_thick=Al_Thickness)

AT (0,0,0) RELATIVE PREVIOUS

COMPONENT H15_5_1 =
Guide_gravity(
    w1=0.03,h1=0.2, w2=0.03, h2=0.2,
    l=L_H15_5,

    R0=gR0, Qc=gQc, alpha=gAlpha,
    m=gMvalue, W=gW)

AT (0,0,Al_Thickness+gGap)
RELATIVE PREVIOUS ROTATED
(0,Rh_H15_5,0) RELATIVE PREVIOUS

```

```

COMPONENT H15_5_2 =
Guide_gravity(
    w1=0.03, h1=0.2, w2=0.03, h2=0.2,
    l=L_H15_5,
    R0=gR0, Qc=gQc, alpha=gAlpha,
    m=gMvalue, W=gW)
AT (0,0,L_H15_5+gGap) RELATIVE
PREVIOUS ROTATED (0,Rh_H15_5,0)
RELATIVE PREVIOUS

COMPONENT H15_5_3 =
Guide_gravity(
    w1=0.03, h1=0.2, w2=0.03, h2=0.2,
    l=L_H15_5,
    R0=gR0, Qc=gQc, alpha=gAlpha,
    m=gMvalue, W=gW)
AT (0,0,L_H15_5+gGap) RELATIVE
PREVIOUS ROTATED (0,Rh_H15_5,0)
RELATIVE PREVIOUS

COMPONENT H15_5_4 =
Guide_gravity(
    w1=0.03, h1=0.2, w2=0.03, h2=0.2,
    l=L_H15_5,
    R0=gR0, Qc=gQc, alpha=gAlpha,
    m=gMvalue, W=gW)
AT (0,0,L_H15_5+gGap) RELATIVE
PREVIOUS ROTATED (0,Rh_H15_5,0)
RELATIVE PREVIOUS

COMPONENT H15_5_5 =
Guide_gravity(
    w1=0.03, h1=0.2, w2=0.03, h2=0.2,
    l=L_H15_5,
    R0=gR0, Qc=gQc, alpha=gAlpha,
    m=gMvalue, W=gW)
AT (0,0,L_H15_5+gGap) RELATIVE
PREVIOUS ROTATED (0,Rh_H15_5,0)
RELATIVE PREVIOUS

COMPONENT H15_5_Out =
Al_window(win_thick=Al_Thickness)
AT (0,0,L_H15_5+gGap) RELATIVE
PREVIOUS

/* gap 330 m (VTE) */

/* ***** after the VTE ***** */

/* H15-6: L=11.473 m in 12 elements
Rh=2700 m */

COMPONENT H15_6 = Arm()
AT (0,0,0.330) RELATIVE H15_5_Out

/*
COMPONENT Mon_6_xy =
Monitor_nD(
    xwidth=0.03, yheight=0.2,
    options='x y, parallel, per cm2, slit')
AT (0,0,0) RELATIVE PREVIOUS

COMPONENT Mon_6_dxdy =
Monitor_nD(
    xwidth=0.03, yheight=0.2,
    options='dx dy, all auto, parallel, per
cm2, slit')
AT (0,0,0) RELATIVE PREVIOUS

COMPONENT Mon_6_Phic =
Monitor_nD(
    xwidth=0.03, yheight=0.2,
    options='x y dx dy, all auto, parallel,
per cm2, capture, slit')
AT (0,0,0) RELATIVE PREVIOUS

COMPONENT Mon_6_L = Monitor_nD(
    xwidth=0.03, yheight=0.2,
    options='lambda, limits=[1 21] bins=20,
parallel, per cm2, slit')
AT (0,0,0) RELATIVE PREVIOUS

*/

COMPONENT H15_6_In =
Al_window(win_thick=Al_Thickness)
AT (0,0,0) RELATIVE PREVIOUS

COMPONENT H15_6_1 =
Guide_gravity(
    w1=0.03, h1=0.2, w2=0.03, h2=0.2,
    l=L_H15_6,
    R0=gR0, Qc=gQc, alpha=gAlpha,
    m=gMvalue, W=gW)
AT (0,0,Al_Thickness+gGap) RELATIVE
PREVIOUS ROTATED (0,Rh_H15_6,0)
RELATIVE PREVIOUS

COMPONENT H15_6_2 =
Guide_gravity(
    w1=0.03, h1=0.2, w2=0.03, h2=0.2,
    l=L_H15_6,
    R0=gR0, Qc=gQc, alpha=gAlpha,
    m=gMvalue, W=gW)
AT (0,0,L_H15_6+gGap) RELATIVE
PREVIOUS ROTATED (0,Rh_H15_6,0)
RELATIVE PREVIOUS

COMPONENT H15_6_3 =
Guide_gravity(
    w1=0.03, h1=0.2, w2=0.03, h2=0.2,
    l=L_H15_6,
    R0=gR0, Qc=gQc, alpha=gAlpha,
    m=gMvalue, W=gW)
AT (0,0,L_H15_6+gGap) RELATIVE
PREVIOUS ROTATED (0,Rh_H15_6,0)
RELATIVE PREVIOUS

COMPONENT H15_6_4 =
Guide_gravity(
    w1=0.03, h1=0.2, w2=0.03, h2=0.2,
    l=L_H15_6,
    R0=gR0, Qc=gQc, alpha=gAlpha,
    m=gMvalue, W=gW)
AT (0,0,L_H15_6+gGap) RELATIVE
PREVIOUS ROTATED (0,Rh_H15_6,0)
RELATIVE PREVIOUS

```

```

RELATIVE PREVIOUS
    w1=0.03, h1=0.2, w2=0.03, h2=0.2,
    l=L_H15_6,

    R0=gR0, Qc=gQc, alpha=gAlpha,
    m=gMvalue, W=gW)

COMPONENT H15_6_5 =
Guide_gravity(
    w1=0.03, h1=0.2, w2=0.03, h2=0.2,
    l=L_H15_6,

    R0=gR0, Qc=gQc, alpha=gAlpha,
    m=gMvalue, W=gW)

AT (0,0,L_H15_6+gGap) RELATIVE
PREVIOUS ROTATED (0,Rh_H15_6,0)
RELATIVE PREVIOUS

COMPONENT H15_6_6 =
Guide_gravity(
    w1=0.03, h1=0.2, w2=0.03, h2=0.2,
    l=L_H15_6,

    R0=gR0, Qc=gQc, alpha=gAlpha,
    m=gMvalue, W=gW)

AT (0,0,L_H15_6+gGap) RELATIVE
PREVIOUS ROTATED (0,Rh_H15_6,0)
RELATIVE PREVIOUS

COMPONENT H15_6_7 =
Guide_gravity(
    w1=0.03, h1=0.2, w2=0.03, h2=0.2,
    l=L_H15_6,

    R0=gR0, Qc=gQc, alpha=gAlpha,
    m=gMvalue, W=gW)

AT (0,0,L_H15_6+gGap) RELATIVE
PREVIOUS ROTATED (0,Rh_H15_6,0)
RELATIVE PREVIOUS

COMPONENT H15_6_8 =
Guide_gravity(
    w1=0.03, h1=0.2, w2=0.03, h2=0.2,
    l=L_H15_6,

    R0=gR0, Qc=gQc, alpha=gAlpha,
    m=gMvalue, W=gW)

AT (0,0,L_H15_6+gGap) RELATIVE
PREVIOUS ROTATED (0,Rh_H15_6,0)
RELATIVE PREVIOUS

COMPONENT H15_6_9 =
Guide_gravity(
    w1=0.03, h1=0.2, w2=0.03, h2=0.2,
    l=L_H15_6,

    R0=gR0, Qc=gQc, alpha=gAlpha,
    m=gMvalue, W=gW)

AT (0,0,L_H15_6+gGap) RELATIVE
PREVIOUS ROTATED (0,Rh_H15_6,0)
RELATIVE PREVIOUS

COMPONENT H15_6_10 =
Guide_gravity(
    w1=0.03, h1=0.2, w2=0.03, h2=0.2,
    l=L_H15_6,

    R0=gR0, Qc=gQc, alpha=gAlpha,
    m=gMvalue, W=gW)

AT (0,0,L_H15_6+gGap) RELATIVE
PREVIOUS ROTATED (0,Rh_H15_6,0)
RELATIVE PREVIOUS

COMPONENT H15_6_11 =
Guide_gravity(
    w1=0.03, h1=0.2, w2=0.03, h2=0.2,
    l=L_H15_6,

    R0=gR0, Qc=gQc, alpha=gAlpha,
    m=gMvalue, W=gW)

AT (0,0,L_H15_6+gGap) RELATIVE
PREVIOUS ROTATED (0,Rh_H15_6,0)
RELATIVE PREVIOUS

COMPONENT H15_6_12 =
Guide_gravity(
    w1=0.03, h1=0.2, w2=0.03, h2=0.2,
    l=L_H15_6,

    R0=gR0, Qc=gQc, alpha=gAlpha,
    m=gMvalue, W=gW)

AT (0,0,L_H15_6+gGap) RELATIVE
PREVIOUS ROTATED (0,Rh_H15_6,0)
RELATIVE PREVIOUS

COMPONENT H15_6_Out =
Al_window(win_thick=Al_Thickness)

AT (0,0,L_H15_6+gGap) RELATIVE
PREVIOUS

/* gap 0.03 m */

/* H15-7: L=9.973 m in 10 elements
Rh=2700 m */

COMPONENT H15_7 = Arm()
AT (0,0,0.03) RELATIVE H15_6_Out

/*

COMPONENT Mon_7_xy =
Monitor_nD(
    xwidth=0.06, yheight=gH,
    options='x y, parallel, per cm2, slit')
AT (0,0,0) RELATIVE PREVIOUS

COMPONENT Mon_7_dxdy =
Monitor_nD(
    xwidth=0.06, yheight=gH,
    options='dx dy, all auto, parallel, per
cm2, slit')
AT (0,0,0) RELATIVE PREVIOUS

COMPONENT Mon_7_Phic =
Monitor_nD(
    xwidth=0.03, yheight=0.2,
    options='x y dx dy, all auto, parallel,
per cm2, capture, slit')
AT (0,0,0) RELATIVE PREVIOUS

COMPONENT Mon_7_L = Monitor_nD(
    xwidth=0.03, yheight=0.2,
    options='lambda, limits=[1 21] bins=20,
parallel, per cm2, slit')
AT (0,0,0) RELATIVE PREVIOUS

*/

COMPONENT H15_7_In =
Al_window(win_thick=Al_Thickness)
AT (0,0,0) RELATIVE PREVIOUS

```

```

l=L_H15_7,

COMPONENT H15_7_1 =
Guide_gravity(

    w1=0.03,h1=0.2, w2=0.03, h2=0.2,
    l=L_H15_7,

    R0=gR0, Qc=gQc, alpha=gAlpha,
    m=gMvalue, W=gW)

AT (0,0,L_H15_7+gGap) RELATIVE
PREVIOUS ROTATED (0,Rh_H15_7,0)
RELATIVE PREVIOUS

COMPONENT H15_7_2 =
Guide_gravity(

    w1=0.03,h1=0.2, w2=0.03, h2=0.2,
    l=L_H15_7,

    R0=gR0, Qc=gQc, alpha=gAlpha,
    m=gMvalue, W=gW)

AT (0,0,L_H15_7+gGap) RELATIVE
PREVIOUS ROTATED (0,Rh_H15_7,0)
RELATIVE PREVIOUS

COMPONENT H15_7_3 =
Guide_gravity(

    w1=0.03,h1=0.2, w2=0.03, h2=0.2,
    l=L_H15_7,

    R0=gR0, Qc=gQc, alpha=gAlpha,
    m=gMvalue, W=gW)

AT (0,0,L_H15_7+gGap) RELATIVE
PREVIOUS ROTATED (0,Rh_H15_7,0)
RELATIVE PREVIOUS

COMPONENT H15_7_4 =
Guide_gravity(

    w1=0.03,h1=0.2, w2=0.03, h2=0.2,
    l=L_H15_7,

    R0=gR0, Qc=gQc, alpha=gAlpha,
    m=gMvalue, W=gW)

AT (0,0,L_H15_7+gGap) RELATIVE
PREVIOUS ROTATED (0,Rh_H15_7,0)
RELATIVE PREVIOUS

COMPONENT H15_7_5 =
Guide_gravity(

    w1=0.03,h1=0.2, w2=0.03, h2=0.2,
    l=L_H15_7,

    R0=gR0, Qc=gQc, alpha=gAlpha,
    m=gMvalue, W=gW)

AT (0,0,L_H15_7+gGap) RELATIVE
PREVIOUS ROTATED (0,Rh_H15_7,0)
RELATIVE PREVIOUS

COMPONENT H15_7_6 =
Guide_gravity(

    w1=0.03,h1=0.2, w2=0.03, h2=0.2,
    l=L_H15_7,

    R0=gR0, Qc=gQc, alpha=gAlpha,
    m=gMvalue, W=gW)

AT (0,0,L_H15_7+gGap) RELATIVE
PREVIOUS ROTATED (0,Rh_H15_7,0)
RELATIVE PREVIOUS

COMPONENT H15_7_7 =
Guide_gravity(

    w1=0.03,h1=0.2, w2=0.03, h2=0.2,
    l=L_H15_7,

    R0=gR0, Qc=gQc, alpha=gAlpha,
    m=gMvalue, W=gW)

AT (0,0,L_H15_7+gGap) RELATIVE
PREVIOUS ROTATED (0,Rh_H15_7,0)
RELATIVE PREVIOUS

COMPONENT H15_7_8 =
Guide_gravity(

    w1=0.03,h1=0.2, w2=0.03, h2=0.2,
    l=L_H15_7,

    R0=gR0, Qc=gQc, alpha=gAlpha,
    m=gMvalue, W=gW)

AT (0,0,L_H15_7+gGap) RELATIVE
PREVIOUS ROTATED (0,Rh_H15_7,0)
RELATIVE PREVIOUS

COMPONENT H15_7_9 =
Guide_gravity(

    w1=0.03,h1=0.2, w2=0.03, h2=0.2,
    l=L_H15_7,

    R0=gR0, Qc=gQc, alpha=gAlpha,
    m=gMvalue, W=gW)

AT (0,0,L_H15_7+gGap) RELATIVE
PREVIOUS ROTATED (0,Rh_H15_7,0)
RELATIVE PREVIOUS

RELATIVE PREVIOUS

COMPONENT H15_7_10 =
Guide_gravity(

    w1=0.03,h1=0.2, w2=0.03, h2=0.2,
    l=L_H15_7,

    R0=gR0, Qc=gQc, alpha=gAlpha,
    m=gMvalue, W=gW)

AT (0,0,L_H15_7+gGap) RELATIVE
PREVIOUS ROTATED (0,Rh_H15_7,0)
RELATIVE PREVIOUS

COMPONENT H15_7_Out =
Al_window(win_thick=Al_Thickness)

AT (0,0,L_H15_7+gGap) RELATIVE
PREVIOUS

// 9.3e9 down to 6.7e9 capture flux
(28/02/2002) with white beam

/* gap 0.3 m */

/* H15-7: L=5.573 m in 6 elements
Rh=2700 m. Here IN6 position. */

COMPONENT H15_8 = Arm()

AT (0,0,0.15) RELATIVE H15_7_Out

/* ----- IN6
Monochromators GROUP -----
---- */

COMPONENT Cradle = Arm()

AT (0,0,0) RELATIVE H15_8

/* triple-monochromator description:

* 7 blades, vertically focusing RV=3 m,
fixed.

* Each blade is 54 mm width, 29 mm
height. Vertical angle +/- 3 deg.

```

* mosaic 23 to 40 min. Motors 0.012 deg/step

* distance between each crystal ensemble 4 cm

*/

SPLIT COMPONENT Mono1 =
Monochromator_curve d(

RV = RV, NV = 7, NH=1,

z width = 0.054, yheight = 0.029,

DM = 3.355, gap = 0.001,

mosaic = 40, r0=1, t0=1,

reflect="HOPG.rfl",
transmit="HOPG.trm")

AT (0,0, -LMM) RELATIVE Cradle
ROTATED (0,A1/2,0) RELATIVE
Cradle

GROUP IN6Monoks

EXTEND

%{

if (SCATTERED) {
MONND_USER_VALUE(MonokMonito
r, 1, 1); }

%}

COMPONENT Mono2 =
Monochromator_curve d(

RV = RV, NV = 7, NH=1,

z width = 0.054, yheight = 0.029,

DM = 3.355, gap = 0.001,

mosaic = 40, r0=1, t0=1,

reflect="HOPG.rfl",
transmit="HOPG.trm")

AT (0,0, 0) RELATIVE Cradle
ROTATED (0,A2/2,0) RELATIVE
Cradle

GROUP IN6Monoks

EXTEND

%{

if (SCATTERED) {

MONND_USER_VALUE(MonokMonito
r, 1, 2); }

%}

COMPONENT Mono3 =
Monochromator_curve d(

RV = RV, NV = 7, NH=1,

z width = 0.054, yheight = 0.029,

DM = 3.355, gap = 0.001,

mosaic = 40, r0=1, t0=1,

reflect="HOPG.rfl",
transmit="HOPG.trm")

AT (0,0, +LMM) RELATIVE Cradle
ROTATED (0,A3/2,0) RELATIVE
Cradle

GROUP IN6Monoks

EXTEND

%{

if (SCATTERED) {
MONND_USER_VALUE(MonokMonito
r, 1, 3); }

%}

/* sample position direction */

COMPONENT mono_out = Arm()

AT (0,0,0) RELATIVE Cradle
ROTATED (0,A2cradle,0) RELATIVE
Cradle

/* ----- IN6 Suppressor -
----- */

COMPONENT SuppPos = Arm()

AT (0,0,LME-LCE-LCC) RELATIVE
mono_out

COMPONENT Mon_SuppInL =
Monitor_nD(

xwidth = 0.05, yheight = 0.098,

options="lambda, all auto")

AT (0,0,-0.07-0.002) RELATIVE SuppPos

COMPONENT Mon_SuppInT =
Monitor_nD(

xwidth = 0.052, yheight = 0.098,

options="t slit, all auto",
bins=iTO F_CHA_RESOL)

AT (0,0,-0.07-0.001) RELATIVE SuppPos

EXTEND

%{

double Vi=sqrt(vx*vx+vy*vy+vz*vz);

if (iRATIO && Vi) {

/* compress flux/s in to opening time
atan(w/length)/PI/frequency */

/* suppressor time spread */

t = -0.07/Vi+(rand01()-
0.5)/PI/frequency*iRATIO*atan(0.052/0.
14);

p/=
PI*Frequency/iRATIO/atan(0.052/0.14);

}

%}

/* Suppressor Chopper position. */

COMPONENT Suppressor =
FermiChopper(radius=0.07, nu=>
Frequency/iRATIO,

height=0.098, width=0.052, Nslit=1,
R0=0, phase=0,

length=0.012, eff=1, verbose=1)

WHEN (iRATIO > 0)

AT (0,0,0) RELATIVE SuppPos

COMPONENT Mon_SuppOutT =
Monitor_nD(

xwidth = 0.052, yheight = 0.098,

options="auto t slit",
bins=iTO F_CHA_RESOL)

AT (0,0,+0.07+0.001) RELATIVE
SuppPos

/* ----- IN6 Fermi -----
----- */

COMPONENT FermiPos = Arm()

AT (0,0,LME-LCE) RELATIVE
mono_out

COMPONENT FermiM =
FermiChopper(phase=iPHASE,
radius=0.04, nu=-Frequency,

height=0.064, width=0.044, Nslit=200.0,
R0=99,

Qc=(mFC < 1 & mFC ?
mFC*0.02176 : 0.02176), alpha=2.33,
m=mFC, length=0.012, eff=1.0,
verbose=1)

AT (0,0,0) RELATIVE FermiPos

COMPONENT Mon_FermiOutdT =
Monitor_nD(

xwidth = 0.044, yheight = 0.064,

options="auto t slit",
bins=iTOF_CHA_RESOL)

AT (0,0,+0.06+0.001) RELATIVE
FermiPos

/* ----- IN6 Fermi END -
----- */

COMPONENT MonokMonitor =
Monitor_nD(

xwidth = 0.2, yheight = 0.2,

options="user1 limits=[0.5,3.5] bins=9,
auto lambda bins=20, square, per cm2")

AT (0,0,+0.06+0.002) RELATIVE
FermiPos

/* sample position (at 2.1 m from
monoks) */

COMPONENT Mon_SampleInT =
Monitor_nD(

xwidth = 0.05, yheight = 0.05,

options="auto t parallel, per cm2",
bins=iTOF_CHA_RESOL)

AT (0,0,LME-.273) RELATIVE
mono_out

COMPONENT Mon_SampleInXY =
Monitor_nD(

xwidth = 0.2, yheight = 0.1,

options="auto x y parallel, per cm2")

AT (0,0,0) RELATIVE PREVIOUS

/* BEGIN

Sample environment and sample */

COMPONENT sample_pos = Arm()

AT (0,0,LME) RELATIVE mono_out

SPLIT COMPONENT
Sample=Isotropic_Sqw(

radius = radius, thickness=thickness,
yheight = 0.055,

Sqw_coh=Sqw_coh, Sqw_inc=Sqw_inc,
p_interact=0.9

) AT (0, 0, 0) RELATIVE sample_pos

EXTEND

%{

if (!SCATTERED) ABSORB;

%}

COMPONENT M_theta_t_all =
Monitor_nD(

xwidth=2.5, yheight=0.2,

options=opt1,

bins=100)

AT (0,0,0) RELATIVE sample_pos

COMPONENT M_omega_q_all =
Monitor_nD(

xwidth=2.6, yheight=0.2,

options=opt2,

bins=100)

AT (0,0,0) RELATIVE sample_pos

END

NXT_IN8.instr

```

/*****
*****
*****
*
* McStas, neutron ray-tracing package
* Copyright (C) 1997-2010, All
rights reserved
* Risoe National Laboratory,
Roskilde, Denmark
* Institut Laue Langevin, Grenoble,
France
* Instrument: NXT_IN8
*
* %Identification
* Written by: <a
href="mailto:farhi@ill.fr">Emmanuel
Farhi</a>
* Date: 2006
* Origin: <a
href="http://www.ill.fr">ILL
(France)</a>
* Release: McStas 1.12b
* Version: $Revision: 1.6 $
* %INSTRUMENT_SITE: Templates
*
* Template RESCAL type triple-axis
machine (TAS)
*
* %Description
* This instrument is a simple model of
triple-axis spectrometer.
* It is directly illuminated by the
moderator,
* and has flat monochromator and
analyzer. Sample is a vanadium cylinder.
* Default geometry is from IN20@ILL.
*
* Si 111 DM=3.135 AA
* PG 002 DM=3.355 AA (Highly
Oriented Pyrolytic Graphite)
* Ge 311 DM=1.714 AA
*
*
* IN8 with PG
*
* PG 002
DM=3.355 AA
* Cu 200 DM=5.550 AA
* Si 111
DM=3.135 AA
*
*
* IN22 configuration:
* KI=3.84, QM=1.0, EN=0.0,
verbose=1,
* WM=0.15, HM=0.12, NHM=1,
NVM=9, RMV=1,
* WA=0.20, HA=0.10, NHA=11,
NVA=3, RAV=-1, RAH=-1,
* SM=1, SS=1, SA=-1,
* L1=10.0, L2=1.7, L3=1.0, L4=0.8

```

```

*
* Example: mcrun template TAS instr
EN=0.5 QM=1
* Example: mcrun template TAS instr
A1=20.6
*
* %Parameters
* INPUT PARAMETERS:
* KI: Incoming neutron wave vector
[Angs-1]
* KF: Outgoing neutron wave vector
[Angs-1]
* EI: Incoming neutron energy [meV]
* EF: Outgoing neutron energy [meV]
* QH: Measurement QH position in
crystal [rlu]
* QK: Measurement QK position in
crystal [rlu]
* QL: Measurement QL position in
crystal [rlu]
* EN: Energy transfer in crystal [meV]
* QM: Wave vector transfer in crystal
[Angs-1]
* KFIX: Fixed KI or KF value for Rescal
compatibility [Angs-1]
* FX: Fixed KI or KF type for Rescal
compatibility [1:KI,2:KF]
* L1: Source-Monochromator distance
[m]
* L2: Monochromator-Sample distance
[m]
* L3: Sample-Analyzer distance [m]
* L4: Analyzer-detector distance [m]
* SM: Scattering sense of beam from
Monochromator [1:left, -1:right]
* SS: Scattering sense of beam from
Sample [1:left, -1:right]
* SA: Scattering sense of beam from
Analyzer [1:left, -1:right]
* DM: Monochromator d-spacing
[Angs]
* DA: Analyzer d-spacing [Angs]
* RMV: Monochromator vertical
curvature, 0 for flat, -1 for automatic
setting [m]
* RMH: Monochromator horizontal
curvature, 0 for flat, -1 for automatic
setting [m]
* RAV: Analyzer vertical curvature, 0 for
flat, -1 for automatic setting [m]
* RAH: Analyzer horizontal curvature, 0
for flat, -1 for automatic setting [m]
* ETAM: Monochromator mosaic [arc
min]
* ETAA: Analyzer mosaic [arc min]
* ALF1: Horizontal collimation from
Source to Monochromator [arc min]
* ALF2: Horizontal collimation from
Monochromator to Sample A [arc min]
* ALF3: Horizontal collimation from
Sample to Analyzer [arc min]
* ALF4: Horizontal collimation from
Analyzer to Detector [arc min]

```

```

* BET1: Vertical collimation from Source
to Monochromator [arc min]
* BET2: Vertical collimation from
Monochromator to Sample A [arc min]
* BET3: Vertical collimation from
Sample to Analyzer [arc min]
* BET4: Vertical collimation from
Analyzer to Detector [arc min]
* AS: Sample lattice parameter A [Angs]
* BS: Sample lattice parameter B
[Angs]
* CS: Sample lattice parameter C
[Angs]
* AA: Angle between lattice vectors B,C
[deg]
* BB: Angle between lattice vectors C,A
[deg]
* CC: Angle between lattice vectors A,B
[deg]
* AX: First reciprocal lattice vector in
scattering plane, X [rlu]
* AY: First reciprocal lattice vector in
scattering plane, Y [rlu]
* AZ: First reciprocal lattice vector in
scattering plane, Z [rlu]
* BX: Second reciprocal lattice vector in
scattering plane, X [rlu]
* BY: Second reciprocal lattice vector in
scattering plane, Y [rlu]
* BZ: Second reciprocal lattice vector in
scattering plane, Z [rlu]
* A1: Monochromator rotation angle
[deg]
* A2: Monochromator take-off angle
[deg]
* A3: Sample rotation angle [deg]
* A4: Sample take-off angle [deg]
* A5: Analyzer rotation angle [deg]
* A6: Analyzer take-off angle [deg]
* SPEED: Velocity of the NXT
[%]
*
* %Link
* Rescal for Matlab at
http://www.ill.fr/tas/matlab
* %Link
* Restrax at
http://omega.ujf.cas.cz/restrax/
* %End
*****
*****
*****/

DEFINE INSTRUMENT NXT_IN8(
KI=0, KF=0, EI=0, EF=0,
QH=0, QK=0, QL=0,
EN=0, QM=0, KFIX=0, FX=0,
L1=2.284, L2=2.284, L3=0.8, L4=0.3,
SM=1, SS=-1, SA=1,
DM=3.355, DA=3.355,
RMV=-1, RMH=-1, RAV=-1, RAH=-1,
ETAM=30, ETAA=30,
ALF1=60, ALF2=60, ALF3=60,

```



```

ALF4=60,
BET1=120, BET2=120, BET3=120,
BET4=120,
AS=6.28, BS=6.28, CS=6.28,
AA=90, BB=90, CC=90,
AX=1, AY=0, AZ=0,
BX=0, BY=1, BZ=0,
verbose=1,
A1=0, A2=0, A3=0, A4=0, A5=0, A6=0,
SPEED=0
)

DECLARE
%{
struct sample_struct {
    double as, bs, cs;
    double aa, bb, cc;
    double ax, ay, az;
    double bx, by, bz;
} sample;

struct machine_hkl_struct {
    double dm, da;
    double l1, l2, l3, l4;
    double sm, ss, sa;
    double etam, etaa, kfix, fx;
    double alf1, alf2, alf3, alf4;
    double bet1, bet2, bet3, bet4;
    double ki, kf, ei, ef;
    double qh, qk, ql, en;
} machine_hkl;

struct machine_real_struct {
    double a1, a2, a3, a4, a5, a6;
    double rmh, rmv, rah, rav;
    double qm, qs, qt[3];
    char message[256];
} machine_real;

struct machine_real_struct qhkl2angles(
    struct sample_struct sample,
    struct machine_hkl_struct machine_hkl,
    struct machine_real_struct machine_real) {

    /* code from
    TASMAD/t_rlp.F:SEIRLP */
    double qhkl[3];
    double alpha[3];
    double a[3];
    double aspv[3][2];
    double cosa[3], sina[3];
    double cosb[3], sinb[3];
    double b[3], c[3], s[4][4];
    double vv[3][3], bb[3][3];
    double arg, cc;
    int i, j, k, l, m, n;
    char liquid_case=1;
    /* transfert parameters to local
    arrays */
    qhkl[0] = machine_hkl.qh; /* HKL
    target */
    qhkl[1] = machine_hkl.qk;
    qhkl[2] = machine_hkl.ql;
    alpha[0] = sample.aa; /* cell angles */
    alpha[1] = sample.bb;

    alpha[2] = sample.cc;
    a[0] = sample.as; /* cell
    parameters */
    a[1] = sample.bs;
    a[2] = sample.cs;
    aspv[0][0] = sample.ax; /* cell axis A */
    aspv[1][0] = sample.ay;
    aspv[2][0] = sample.az;
    aspv[0][1] = sample.bx; /* cell axis B
    */
    aspv[1][1] = sample.by;
    aspv[2][1] = sample.bz;

    /* default return values */
    strcpy(machine_real.message, "");
    machine_real.a3 = machine_real.a4 =
    0;
    machine_real.a1 = machine_real.a5 =
    0;

    /* if using HKL positioning in crystal
    (QM=0) */
    if (machine_real.qm <= 0) {
        liquid_case = 0;
        /* compute reciprocal cell */
        for (i=0; i<3; i++)
            if (a[i] <= 0)
                sprintf(machine_real.message, "Lattice
                parameters a[%i]=%g", i, a[i]);
        else {
            a[i] /= 2*PI;
            alpha[i] = DEG2RAD;
            cosa[i] = cos(alpha[i]);
            sina[i] = sin(alpha[i]);
        }
        cc =
        cosa[0]*cosa[0]+cosa[1]*cosa[1]+cosa[2]*
        cosa[2]; /* norm */
        cc = 1 + 2*cosa[0]*cosa[1]*cosa[2] -
        cc;
        if (cc <= 0)
            sprintf(machine_real.message, "Lattice
            angles (AA,BB,CC) cc=%g", cc);
        else cc = sqrt(cc);

        if (strlen(machine_real.message))
            return machine_real;

        /* compute bb */
        j=1; k=2;
        for (i=0; i<3; i++) {
            b[i] = sina[i]/(a[i]*cc);
            cosb[i] = (cosa[j]*cosa[k] -
            cosa[i])/(sina[j]*sina[k]);
            sinb[i] = sqrt(1 - cosb[i]*cosb[i]);
            j=k; k=i;
        }
        bb[0][0] = b[0];
        bb[1][0] = 0;
        bb[2][0] = 0;
        bb[0][1] = b[1]*cosb[2];
        bb[1][1] = b[1]*sinb[2];
        bb[2][1] = 0;
        bb[0][2] = b[2]*cosb[1];
        bb[1][2] = b[2]*sinb[1]*cosa[0];
        bb[2][2] = 1/a[2];

        /* compute vv */
        for (k=0; k<3; k++)
            for (i=0; i<3; i++) vv[k][i] = 0;

        for (k=0; k<2; k++)
            for (i=0; i<3; i++)
                for (j=0; j<3; j++)
                    vv[k][i] += bb[i][j]*aspv[j][k];

        for (m=2; m>=1; m--)
            for (n=0; n<3; n++) {
                i = (int)fmod(m+1,3); j =
                (int)fmod(m+2,3);
                k = (int)fmod(n+1,3); l =
                (int)fmod(n+2,3);
                vv[m][n] = vv[i][k]*vv[j][l] -
                vv[i][l]*vv[j][k];
            }

        for (i=0; i<3; i++) { /* compute
        norm(vv) */
            c[i]=0;
            for (j=0; j<3; j++)
                c[i] += vv[i][j]*vv[i][j];
            if (c[i]>0) c[i] = sqrt(c[i]);
            else {
                sprintf(machine_real.message,
                "Vectors A and B, c[%i]=%g", i, c[i]);
                return machine_real;
            }
        }

        for (i=0; i<3; i++) /* normalize vv */
            for (j=0; j<3; j++)
                vv[j][i] /= c[i];

        for (i=0; i<3; i++) /* compute S */
            for (j=0; j<3; j++) {
                s[i][j] = 0;
                for (k=0; k<3; k++)
                    s[i][j] += vv[i][k]*bb[k][j];
            }
        s[3][3]=1;
        for (i=0; i<3; i++) s[3][i]=s[i][3]=0;

        /* compute q modulus and
        transverse component */
        machine_real.qs = 0;
        for (i=0; i<3; i++) {
            machine_real.qt[i] = 0;
            for (j=0; j<3; j++)
                machine_real.qt[i] += qhkl[j]*s[i][j];
            machine_real.qs +=
            machine_real.qt[i]*machine_real.qt[i];
        }
        if (machine_real.qs > 0)
            machine_real.qm =
            sqrt(machine_real.qs);
            else sprintf(machine_real.message,
            "Q modulus too small QM^2=%g",
            machine_real.qs);
        } else {
            machine_real.qs =
            machine_real.qm*machine_real.qm;
        }
        /* endif qm <= 0

```



```

*****
***** */

/* positioning of monochromator and
analyser */
arg =
PI/machine_hkl.dm/machine_hkl.ki;
if (fabs(arg > 1))
    sprintf(machine_real.message,
"Monochromator can not reach this KI.
arg=%g", arg);
else {
    if (machine_hkl.dm <= 0 ||
machine_hkl.ki <= 0)
        strcpy(machine_real.message,
"Monochromator DM=0 or KI=0.");
    else
        machine_real.a1 =
asin(arg)*RAD2DEG;
        machine_real.a1 *=
machine_hkl.sm;
    }
    machine_real.a2=2*machine_real.a1;

    arg =
PI/machine_hkl.da/machine_hkl.kf;
    if (fabs(arg > 1))
        sprintf(machine_real.message,
"Analyzer can not reach this KF.
arg=%g", arg);
    else {
        if (machine_hkl.da <= 0 ||
machine_hkl.kf <= 0)
            strcpy(machine_real.message,
"Analyzer DA=0 or KF=0.");
        else
            machine_real.a5 =
asin(arg)*RAD2DEG;
            machine_real.a5 *= machine_hkl.sa;
        }
        machine_real.a6=2*machine_real.a5;
        if (strlen(machine_real.message))
            return machine_real;

/* code from
TASMAD/t_conv.F:SAM_CASE */
arg =
(machine_hkl.ki*machine_hkl.ki +
machine_hkl.kf*machine_hkl.kf -
machine_real.qs)
/
(2*machine_hkl.ki*machine_hkl.kf);
if (fabs(arg) < 1)
    machine_real.a4 =
RAD2DEG*acos(arg);
else
    sprintf(machine_real.message, "Q
modulus too big. Can not close triangle.
arg=%g", arg);
    machine_real.a4 *= machine_hkl.ss;

    if (liquid_case) { /* compute a3 in
crystals */
        machine_real.a3 =
-
atan2(machine_real.q[1], machine_real.q

```

```

t[0])
    -acos(
(machine_hkl.kf*machine_hkl.kf-
machine_real.qs-
machine_hkl.ki*machine_hkl.ki)
/(-
2*machine_real.qm*machine_hkl.ki));
    machine_real.a3 *=
RAD2DEG*(machine_real.a4 > 0 ? 1 : -1
);
    }

    return machine_real;
}
*/
/* end of DECLARE */

INITIALIZE
%{
double Vi, Vf;
char anglemode = 0;

if (KFIX && FX) {
    if (FX == 1) KI = KFIX;
    else if (FX == 2) KF = KFIX;
}

/* determine neutron energy from input
*/
if (KI && !EI) {
    Vi = K2V*fabs(KI);
    EI = VS2E*Vi*Vi;
}
if (KF && !EF) {
    Vf = K2V*fabs(KF);
    EF = VS2E*Vf*Vf;
}

machine_real.a1 = A1;
machine_real.a2 = A2;
machine_real.a3 = A3;
machine_real.a4 = A4;
machine_real.a5 = A5;
machine_real.a6 = A6;

if (A1 || A2 || A3 || A4 || A5 || A6)
    anglemode=1;

if (!anglemode) {
    if (!EI && !EF)
        exit(fprintf(stderr,
"%s: ERROR: neutron beam
energy is not defined (EI, EF, KI, KF)\n",
NAME_CURRENT_COMP));

/* energy conservation */
if (EI)
    EF = EI - EN;
else if (EF)
    EI = EF + EN;

/* determine remaining neutron
energies */
if (!KI && EI) {
    Vi = SE2V*sqrt(EI);
    KI = V2K*Vi;
}

```

```

if (!KF && EF) {
    Vf = SE2V*sqrt(EF);
    KF = V2K*Vf;
}

if (!QM && !QH && !QK && !QL)
    exit(fprintf(stderr,
"%s: ERROR: No Q transfer
defined (QM, QH, QK, QL)\n",
NAME_CURRENT_COMP));
}

/* transfert sample parameters */
sample.aa = AA;
sample.bb = BB;
sample.cc = CC;
sample.as = AS;
sample.bs = BS;
sample.cs = CS;
sample.ax = AX;
sample.ay = AY;
sample.az = AZ;
sample.bx = BX;
sample.by = BY;
sample.bz = BZ;

/* transfert target parameters */
machine_hkl.ki = KI;
machine_hkl.kf = KF;
machine_hkl.ei = EI;
machine_hkl.ef = EF;
machine_hkl.qh = QH;
machine_hkl.qk = QK;
machine_hkl.ql = QL;
machine_hkl.en = EN;
machine_real.qm = QM;

if (verbose) {
    printf("%s: Detailed TAS
configuration\n",
NAME_CURRENT_COMP);
    printf("Incoming beam: EI=%g
[meV] KI=%g [Angs-1] Vi=%g
[m/s]\n", EI, KI, Vi);
    printf("Outgoing beam: EF=%g
[meV] KF=%g [Angs-1] Vf=%g
[m/s]\n", EF, KF, Vf);
}

/* transfert machine parameters */
machine_hkl.l1 = L1;
machine_hkl.l2 = L2;
machine_hkl.l3 = L3;
machine_hkl.l4 = L4;
machine_hkl.sm = SM;
machine_hkl.ss = SS;
machine_hkl.sa = SA;
machine_hkl.dm = DM;
machine_hkl.da = DA;
machine_real.rmv = RMV;
machine_real.rmh = RMH;
machine_real.rav = RAV;
machine_real.rah = RAH;
machine_hkl.e.tam = ETAM;
machine_hkl.e.taa = ETAA;
machine_hkl.alf1 = ALF1;
machine_hkl.alf2 = ALF2;

```

```

machine_hkl.alf3=ALF3;
machine_hkl.alf4=ALF4;
machine_hkl.be1=BET1;
machine_hkl.be2=BET2;
machine_hkl.be3=BET3;
machine_hkl.be4=BET4;

/* geometry tests w/r to collimator
lengths */
if (machine_hkl.l1 <= 1)
    exit(fprintf(stderr, "%s: ERROR: L1
too short. Min=5.34\n",
NAME_CURRENT_COMP));

if (machine_hkl.l2 <= 0.35)
    exit(fprintf(stderr, "%s: ERROR: L2
too short. Min=0.35\n",
NAME_CURRENT_COMP));

if (machine_hkl.l3 <= 0.40)
    exit(fprintf(stderr, "%s: ERROR: L3
too short. Min=0.40\n",
NAME_CURRENT_COMP));

if (machine_hkl.l4 <= 0.24)
    exit(fprintf(stderr, "%s: ERROR: L4
too short. Min=0.24\n",
NAME_CURRENT_COMP));

if (!anglemode) {
    machine_real = qhkl2angles(sample,
machine_hkl, machine_real);
    if (strlen(machine_real.message))
        exit(fprintf(stderr, "%s: ERROR: %s
[qhkl2angles]\n",
NAME_CURRENT_COMP,
machine_real.message));
}

/* compute optimal curvatures */
double L;
L = 1/(1/L1+1/L2);
if (RMV < 0) machine_real.rmv =
2*L*sin(DEG2RAD*machine_real.a1);
if (RMH < 0) machine_real.rmh =
2*L/sin(DEG2RAD*machine_real.a1);
L = 1/(1/L3+1/L4);
if (RAV < 0) machine_real.rav =
2*L*sin(DEG2RAD*machine_real.a5);
if (RAH < 0) machine_real.rah =
2*L/sin(DEG2RAD*machine_real.a5);

if (verbose) {
    printf("Transfert: EN=%g [meV]
QM=%g [Angs-1]\n", EN,
machine_real.qm);
    printf("Angles: A1=%g A2=%g
A3=%g A4=%g A5=%g A6=%g
[deg]\n",
machine_real.a1, machine_real.a2,
machine_real.a3, machine_real.a4,
machine_real.a5, machine_real.a6);
    printf("Monochromator: DM=%g
[Angs] RMH=%g [m] RMV=%g [m]
%s\n",
machine_hkl.dm, machine_real.rmh,
machine_real.rmv,

```

```

(machine_real.rmh &&
!machine_real.rmv ? "flat" : "curved"));
printf("Analyzer: DA=%g [Angs]
RAH=%g [m] RAV=%g [m] %s\n",
machine_hkl.da, machine_real.rah,
machine_real.rav,
(machine_real.rah &&
!machine_real.rav ? "flat" : "curved"));
}
/* end of INITIALIZE */

```

```

TRACE
/* Source description */

/* a flat constant source */
COMPONENT Source = Source_gen(
    radius = 0.10,
    dist = machine_hkl.l1,
    xw = 0.1, yh = 0.08,
    E0 = machine_hkl.ei,
    dE = machine_hkl.ei*0.03)
AT (0,0,0) ABSOLUTE

```

COMPONENT
nxt=NXTAS(A1=A1,A2=A2,A3=
A3,A4=A4,A5=A5,A6=A6,SPEE
D=SPEED)
AT (0,0,0) ABSOLUTE

```

COMPONENT SC1 = Collimator_linear(
    xmin = -0.08/2, ymin = -0.11/2,
    xmax = 0.08/2, ymax = 0.11/2,
    len = 1,
    divergence=ALF1,
    divergenceV=BET1)
AT (0, 0, (machine_hkl.l1-1)/2)
RELATIVE Source

```

```

COMPONENT Mono_Cradle = Arm()
AT (0, 0, machine_hkl.l1) RELATIVE
Source
ROTATED (0, machine_real.a1, 0)
RELATIVE Source

```

```

SPLIT COMPONENT PG1Xtal =
Monochromator_curved(
    width = 0.10,
    height = 0.12,
    NH=1, NV=9,
    RV=machine_real.rmv,
    RH=machine_real.rmh,
    mosaic = machine_hkl.e.tam, mosaicv =
machine_hkl.e.tam,
    r0 = 0.7, DM=machine_hkl.dm)
AT (0, 0, 0) RELATIVE Mono_Cradle

```

```

/* on mono, pointing
towards sample */
COMPONENT Mono_Out = Arm()
AT (0,0,0) RELATIVE Mono_Cradle
ROTATED (0, machine_real.a2, 0)
RELATIVE Source

```

```

COMPONENT SC2 = Collimator_linear(
    xmin = -0.04/2, ymin = -0.07/2,

```

```

xmax = 0.04/2, ymax = 0.07/2,
len = 0.35,
divergence=ALF2,
divergenceV=BET2)
AT (0, 0, (machine_hkl.l2-0.35)/2)
RELATIVE Mono_Out

```

```

COMPONENT Sample_Cradle = Arm()
AT (0, 0, machine_hkl.l2) RELATIVE
Mono_Out
ROTATED (0, machine_real.a3, 0)
RELATIVE Mono_Out

```

```

SPLIT COMPONENT Sample =
V_sample(
    radius_i = 0.0, radius_o = 0.0064, h =
0.0254,
    focus_xw = 0.06, focus_yh=0.12, pack =
1,
    target_index=2)
AT (0,0,0) RELATIVE Sample_Cradle

```

```

COMPONENT Sample_Out = Arm() /*
this is the sample-ana axis */
AT (0,0,0) RELATIVE Sample_Cradle
ROTATED (0, machine_real.a4, 0)
RELATIVE Mono_Out

```

```

COMPONENT SC3 = Collimator_linear(
    xmin = -0.06/2, ymin = -0.12/2,
    xmax = 0.06/2, ymax = 0.12/2,
    len = 0.40,
    divergence=ALF3,
    divergenceV=BET3)
AT (0, 0, (machine_hkl.l3-0.40)/2)
RELATIVE Sample_Out

```

```

COMPONENT Ana_Cradle = Arm()
AT (0, 0, machine_hkl.l3) RELATIVE
Sample_Out
ROTATED (0, machine_real.a5, 0)
RELATIVE Sample_Out

```

```

SPLIT COMPONENT PG2Xtal =
Monochromator_curved(
    width = 0.10,
    height = 0.12,
    NH=1, NV=9,
    RV=machine_real.rav,
    RH=machine_real.rah,
    mosaic = machine_hkl.e.taa, mosaicv =
machine_hkl.e.taa,
    r0 = 0.7, DM=machine_hkl.da)
AT (0, 0, 0) RELATIVE Ana_Cradle

```

```

COMPONENT Ana_Out = Arm() /*
this is the sample-ana axis */
AT (0,0,0) RELATIVE Ana_Cradle
ROTATED (0, machine_real.a6, 0)
RELATIVE Sample_Out

```

```

COMPONENT SC4 = Collimator_linear(
    xmin = -0.06/2, ymin = -0.12/2,
    xmax = 0.06/2, ymax = 0.12/2,
    len = 0.24,
    divergence=ALF4,
    divergenceV=BET4)

```

AT (0, 0, (machine_hkl14-0.24)/2)
RELATIVE Ana_Out

/* vertical 3He Detector */
COMPONENT He3H = PSD_monitor(
xmin = -0.025400, xmax = 0.025400,

ymin = -0.042850, ymax = 0.042850,
nx=20, ny=20, filename='He3H.psd')
AT (0, 0, machine_hkl14) RELATIVE
Ana_Out

END

Annexe 4 : Programmes embarqués. nxTAS.nxc

```
//
// Spectrometer TAS with the LEGO
// Mindstorms NXT
//
// 2010, Institut Laue Langevin.

float seekparam(float a, string tempa,
byte handle) // fonction allant
automatiquement à la ligne et enregistrer
les valeurs des paramètres ai.
{
fseek(handle, 2, SEEK_END); //ligne
suivante

fseek(handle, 3, SEEK_CUR); //avance
de 3 caractères

fgets(tempa, 5, handle); //copie les 5
premiers bytes dans tempa

a = StrToNum(tempa); //Converti le
string tempa en float a

return(a);
}

float norm(float a, float b) // fonction
faisant la moyenne entre a et b, avec a
pondéré par 2 et retournant cette
valeur.
{
a=(2*a+b)/4;

return(a);
}

task main()
{
int i=1;

int j=10;

int k=0;

string tempspeed;

int new=0; //si nouvelles coordonnées
détectés 1, sinon 0

float buteeC;

float butee2C;

float buteeB;

float butee2B;

float buteeA;

float butee2A;

float temp;

float speed;

int x;

string test;

float a1;

string tempa1;

float a2;

string tempa2;

float a3;

string tempa3;

float a4;

string tempa4;

float a5;

string tempa5;

float a6;

string tempa6;

float a2_corrected;

float a4_corrected;

float a6_corrected;

int flag_nothing=0;

int flag_mAmoved=0;

int flag_mBmoved=0;

int flag_mCmoved=0;

float a2_before;

float a4_before;

float a6_before;

while(true){
TextOut(0, LCD_LINE1, "
");
TextOut(0, LCD_LINE2, "
");
TextOut(0, LCD_LINE3, "
");
TextOut(0, LCD_LINE4, "
");
TextOut(0, LCD_LINE5, "
");
TextOut(0, LCD_LINE6, "
");
}
```

```

TextOut(0, LCD_LINE7, "
");

TextOut(0, LCD_LINE8, "
");

TextOut(20, LCD_LINE1, "nxTAS
v1.0");

TextOut(17, LCD_LINE2, "config file");

TextOut(10, LCD_LINE3, "made by
McStas");

NumOut(22, LCD_LINE5, a1-30);
//affiche a1

TextOut(2, LCD_LINE5, "A1=");

NumOut(75, LCD_LINE5, a2-60);
//affiche a2

TextOut(55, LCD_LINE5, "A2=");

NumOut(22, LCD_LINE6, a3+10);
//affiche a3

TextOut(2, LCD_LINE6, "A3=");

NumOut(75, LCD_LINE6, a4+20);
//affiche a4

TextOut(55, LCD_LINE6, "A4=");

NumOut(22, LCD_LINE7, a5+80);
//affiche a5

TextOut(2, LCD_LINE7, "A5=");

NumOut(75, LCD_LINE7, a6+160);
//affiche a6

TextOut(55, LCD_LINE7, "A6=");

NumOut(40, LCD_LINE8, speed); //
affiche speed

TextOut(2, LCD_LINE8, "speed=");

//////////////////////////////// Data polling
////////////////////////////////

i=i-1;
if(i==0){
byte handle = fopen("nxtas.txt", "r");

fseek(handle, 10, SEEK_SET); //
débute au début de la première ligne

fgets(test, 30, handle);

fseek(handle, 2, SEEK_END); // ligne
suivante

fgets(test, 30, handle);

fseek(handle, 2, SEEK_END); // le '/'
implique un saut de ligne

fgets(test, 30, handle);

fseek(handle, 2, SEEK_END); //ligne
suivante

fgets(test, 30, handle);

a1=seekparam(a1, tempa1, handle);

a2=seekparam(a2, tempa2, handle);

a3=seekparam(a3, tempa3, handle);

a4=seekparam(a4, tempa4, handle);

a5=seekparam(a5, tempa5, handle);

a6=seekparam(a6, tempa6, handle);

fseek(handle, 2, SEEK_END); //saut
de ligne entre les paramètres angulaires
et les paramètres de vitesse

fgets(tempspeed, 30, handle);

fseek(handle, 2, SEEK_END);

fseek(handle, 6, SEEK_CUR);

fgets(tempspeed, 3, handle);

speed = StrToNum(tempspeed); //
vitesse des moteurs

fclose(handle);

a1=30+a1; //butée
a2=60+a2;
a3=a3-10;
a4=a4-20;
a5=a5-80;
a6=a6-160;

i=3;
}

//////////////////////////////// Data analysis
////////////////////////////////

if (speed<50){ //correction de la vitesse
speed=80;
}

if (speed>100){
speed=100;
}

if (a1!=(a2/2)){ //correction des
paramètres ai rentrés.
a1=norm(a1,a2);
a2=2*a1;

```

	Rotate Motor(OUT_B,speed,- MotorTachoCount(OUT_B));	Reset TachoCount(OUT_A);
}		if(((MotorTachoCount(OUT_B)!=0))){ //asservissement
	Reset TachoCount(OUT_B);}	
if (a3!=(a4/2)){		Rotate Motor(OUT_B,speed,- MotorTachoCount(OUT_B));
a3=norm(a3,a4);	Rotate Motor(OUT_A,speed,-a2/4);	Reset TachoCount(OUT_B);}
a4=2*a3;	Reset TachoCount(OUT_A);	
}	if(((MotorTachoCount(OUT_B)!=0))){ //asservissement	
	Rotate Motor(OUT_B,speed,- MotorTachoCount(OUT_B));	Rotate Motor(OUT_A,speed,-a2/4);
if (a5!=(a6/2)){	Reset TachoCount(OUT_B);}	Reset TachoCount(OUT_A);
a5=norm(a5,a6);		if(((MotorTachoCount(OUT_B)!=0))){ //asservissement
a6=2*a5;		
}	Rotate Motor(OUT_A,speed,-a2/4);	Rotate Motor(OUT_B,speed,- MotorTachoCount(OUT_B));
	Reset TachoCount(OUT_A);	Reset TachoCount(OUT_B);
	if(((MotorTachoCount(OUT_B)!=0))){ //asservissement	}
	Rotate Motor(OUT_B,speed,- MotorTachoCount(OUT_B));	
//////////////////// Moving ////////////////////	Reset TachoCount(OUT_B);}	}
if (new==1){	Rotate Motor(OUT_A,speed,-a2/4);	else {
	Reset TachoCount(OUT_A);	Rotate Motor(OUT_A,speed,-a2/2);
if((a2)>20){	if(((MotorTachoCount(OUT_B)!=0))){ //asservissement	Reset TachoCount(OUT_A);
Reset TachoCount(OUT_A);	Rotate Motor(OUT_B,speed,- MotorTachoCount(OUT_B));	if(((MotorTachoCount(OUT_B)!=0))){ //asservissement
Rotate Motor(OUT_A,speed,-a2/4); // le coefficient multiplicatif appliqué est dû à la configuration du système d'engrenage.	Reset TachoCount(OUT_B);}	Rotate Motor(OUT_B,speed,- MotorTachoCount(OUT_B));
Reset TachoCount(OUT_A);		Reset TachoCount(OUT_B);
if(((MotorTachoCount(OUT_B)!=0))){ //asservissement	Rotate Motor(OUT_A,speed,-a2/4);	}
Rotate Motor(OUT_B,speed,- MotorTachoCount(OUT_B));	Reset TachoCount(OUT_A);	Rotate Motor(OUT_A,speed,-a2/2);
Reset TachoCount(OUT_B); }	if(((MotorTachoCount(OUT_B)!=0))){ //asservissement	Reset TachoCount(OUT_A);
	Rotate Motor(OUT_B,speed,- MotorTachoCount(OUT_B));	if(((MotorTachoCount(OUT_B)!=0))){ //asservissement
Rotate Motor(OUT_A,speed,-a2/4);	Reset TachoCount(OUT_B);}	Rotate Motor(OUT_B,speed,- MotorTachoCount(OUT_B));
Reset TachoCount(OUT_A);		Reset TachoCount(OUT_B);
if(((MotorTachoCount(OUT_B)!=0))){ //asservissement	Rotate Motor(OUT_A,speed,-a2/4);	}
		Rotate Motor(OUT_A,speed,-a2/2);

```

ResetTachoCount(OUT_A);

if(((MotorTachoCount(OUT_B)!=0))){
//asservissement

RotateMotor(OUT_B,speed, -
MotorTachoCount(OUT_B));

ResetTachoCount(OUT_B);
}

RotateMotor(OUT_A,speed,-a2/2);

ResetTachoCount(OUT_A);

if(((MotorTachoCount(OUT_B)!=0))){
//asservissement

RotateMotor(OUT_B,speed, -
MotorTachoCount(OUT_B));

ResetTachoCount(OUT_B);
}
}

ResetTachoCount(OUT_B);

RotateMotor(OUT_B,speed,-a4/4); //le
coefficient multiplicatif appliqué est dû à
la configuration du système d'engrenage.

ResetTachoCount(OUT_B);

RotateMotor(OUT_B,speed,-a4/4);

ResetTachoCount(OUT_B);

RotateMotor(OUT_B,speed,-a4/4);

ResetTachoCount(OUT_B);

RotateMotor(OUT_B,speed,-a4/4);

ResetTachoCount(OUT_B);

```

```

ResetTachoCount(OUT_C);

RotateMotor(OUT_C,30,(a6)/2);

ResetTachoCount(OUT_C);

flag_mAmoved=1;

flag_mBmoved=1;

flag_mCmoved=1;

a2_before=a2;
a4_before=a4;
a6_before=a6;

Wait(SEC_1);

ResetTachoCount(OUT_A);

ResetTachoCount(OUT_B);

ResetTachoCount(OUT_C);

PlayTone(500,500); // Positionnement fini

new=0;
}

if(j>0){

if(((MotorTachoCount(OUT_A)!=0))){
//asservissement

RotateMotor(OUT_A,speed, -
MotorTachoCount(OUT_A));

ResetTachoCount(OUT_A);
}

if(((MotorTachoCount(OUT_B)!=0))){
//asservissement

```

```

RotateMotor(OUT_B,speed, -
MotorTachoCount(OUT_B));

ResetTachoCount(OUT_B);
}

if(((MotorTachoCount(OUT_C)!=0))){
//asservissement

RotateMotor(OUT_C,30,
MotorTachoCount(OUT_C));

ResetTachoCount(OUT_C);
}

j=j-1;
}

if(a6!=a6_before||a4!=a4_before||a2!=a2_
before){

j=10;

new=1; //nouvelles coordonnées

buteeC=100;

buteeB=100;

buteeA=100;

butee2B=200;

butee2A=200;

butee2C=200;

for (k=0;k<40;k++){ //retour a la
position de base A2=-50;A4=38;
A6=152;

if((butee2A>5)|| (butee2A<5)){

RotateMotor(OUT_A,80, 20);

```

```
if(k%2==0){ //détecte la position du
moteur 1 oscillation sur 2.
```

```
butee2A=MotorTachoCount(OUT_A); //
butee2A=0 ou proche de 0 si le moteur est
en bûté.
```

```
}
```

```
}
```

```
if((butee2B>40)|| (butee2B<-40)){
```

```
RotateMotor(OUT_B,60, -10);
```

```
if(k%2==0){
```

```
butee2B=MotorTachoCount(OUT_B);
```

```
}
```

```
}
```

```
if((butee2C>5)|| (butee2C<-5)){
```

```
RotateMotor(OUT_C,70, 10);
```

```
if(k%2==0){
```

```
butee2C=MotorTachoCount(OUT_C);
```

```
}
```

```
}
```

```
ResetTachoCount(OUT_C);
```

```
ResetTachoCount(OUT_B);
```

```
ResetTachoCount(OUT_A);
```

```
}
```

```
}
```

```
Wait(SEC_1);
```

```
ResetSleepTimer();
```

```
}
```

```
}
```


nxTOF.nxc

```
//
// Spectrometer TAS with the LEGO
// Mindstorms NXT
//
// 2010, Institut Laue Langevin.

float seekparam(float a, string tempa,
byte handle) // fonction allant
au automatiquement à la ligne et enregistre
les valeurs des paramètres ai.
{
fseek(handle, 2, SEEK_END); // ligne
suivante

fseek(handle, 3, SEEK_CUR); // avance
de 3 bytes

fgets(tempa, 5, handle); // copie les 5
bytes suivants dans tempa

a = StrToNum(tempa); // transforme
le string en float

return(a);
}

float norm(float a, float b){ // fonction
faisant la moyenne entre a et b, avec a
pondéré par 2 et retournant cette
valeur.

a=(2*a+b)/4;

return(a);
}

task main()
{
int i=1;

int j=10;

int k=0;

string tempspeed;

int new=0; //si nouvelles coordonnées
détectés 1, sinon 0

int speed = 70;

float buteeC;

float butee2C;

float buteeB;

float butee2B;

float buteeA;

float butee2A;

float temp;

int x;

string test;

float a1;

string tempa1;

float a2;

string tempa2;

float a3;

string tempa3;

float a4;

string tempa4;

float a2_corrected;

float a4_corrected;

float fspeed;

int flag_nothing=0;

int flag_mAmoved=0;

int flag_mBmoved=0;

int flag_mCmoved=0;

float a2_before=0;

float a3_before=0;

float fspeed_before=0;

while(true){

TextOut(0, LCD_LINE1, "
");

TextOut(0, LCD_LINE2, "
");

TextOut(0, LCD_LINE3, "
");

TextOut(0, LCD_LINE4, "
");

TextOut(0, LCD_LINE5, "
");

TextOut(0, LCD_LINE6, "
");

TextOut(0, LCD_LINE7, "
");

TextOut(0, LCD_LINE8, "
");

TextOut(20, LCD_LINE1, "nxTOF
v1.0");

TextOut(17, LCD_LINE2, "config file");

TextOut(10, LCD_LINE3, "made by
McStas");

NumOut(22, LCD_LINE5, a1 -40);
//affiche a1

TextOut(2, LCD_LINE5, "A1=");

NumOut(75, LCD_LINE5, a2 -80);
//affiche a2

TextOut(55, LCD_LINE5, "A2=");

NumOut(22, LCD_LINE6, a3); //
```

affiche a3

```
TextOut(2, LCD_LINE6, "A3=");
```

```
NumOut(75, LCD_LINE8, fspeed); //
affiche la vitesse du fermi en pourcent
```

```
TextOut(2, LCD_LINE8, "Fermi
speed=");
```

```
//////////////////// Data polling
////////////////////
```

```
i=i-1;
```

```
if(i==0){
```

```
byte handle = fopen("nxtof.txt", "r");
```

```
fseek(handle, 10, SEEK_SET); // début
de ligne
```

```
fgets(test, 30, handle);
```

```
fseek(handle, 2, SEEK_END); // ligne
suivante
```

```
fgets(test, 30, handle);
```

```
fseek(handle, 2, SEEK_END); // ligne
suivante à cause du / de la date
```

```
fgets(test, 30, handle);
```

```
fseek(handle, 2, SEEK_END); // Ligne
suivante
```

```
fgets(test, 30, handle);
```

```
a1=seekparam(a1, tempa1, handle);
```

```
a2=seekparam(a2, tempa2, handle);
```

```
a3=seekparam(a3, tempa3, handle);
```

```
fseek(handle, 2, SEEK_END);
```

```
fseek(handle, 12, SEEK_CUR);
```

```
fgets(tempspeed, 3, handle);
```

```
fspeed = StrToNum(tempspeed); //
fermi speed
```

```
fclose(handle);
```

```
a1=40+a1; //butee
```

```
a2=80+a2;
```

```
i=3;
```

```
}
```

```
//////////////////// Data analysis
////////////////////
```

```
if (fspeed<10){ //speed correction
```

```
fspeed=10;
```

```
}
```

```
if (fspeed>100){
```

```
fspeed=100;
```

```
}
```

```
if (a1!=(a2/2)){ //parameters correction
```

```
a1=norm(a1,a2);
```

```
a2=2*a1;
```

```
}
```

```
//////////////////// Moving
////////////////////
```

```
if (new==1){
```

```
OnFwd(OUT_B, fspeed); //fermi
```

```
if (a2)>20){
```

```
ResetTachoCount(OUT_A);
```

```
RotateMotor(OUT_A, speed, -a2/4); // le
coefficient multiplicatif appliqué est dû à
la configuration du système d'engrenage.
```

```
ResetTachoCount(OUT_A);
```

```
RotateMotor(OUT_A, speed, -a2/4);
```

```
ResetTachoCount(OUT_A);
```

```
RotateMotor(OUT_A, speed, -a2/4);
```

```
ResetTachoCount(OUT_A);
```

```
RotateMotor(OUT_A, speed, -a2/4);
```

```
ResetTachoCount(OUT_A);
```

```
RotateMotor(OUT_A, speed, -a2/4);
```

```
ResetTachoCount(OUT_A);
```

```
RotateMotor(OUT_A, speed, -a2/4);
```

```
ResetTachoCount(OUT_A);
```

```
RotateMotor(OUT_A, speed, -a2/4);
```

```
ResetTachoCount(OUT_A);
```

```
RotateMotor(OUT_A, speed, -a2/4);
```

```
ResetTachoCount(OUT_A);
```

```
}
```

```
else{
```

```
RotateMotor(OUT_A, speed, -a2/2);
```

```

ResetTachoCount(OUT_A);

RotateMotor(OUT_A,speed,-a2/2);

ResetTachoCount(OUT_A);

RotateMotor(OUT_A,speed,-a2/2);

ResetTachoCount(OUT_A);

RotateMotor(OUT_A,speed,-a2/2);

ResetTachoCount(OUT_A);

}

ResetTachoCount(OUT_C);

RotateMotor(OUT_C,speed,(a3-
a3_before)/4);

ResetTachoCount(OUT_C);

RotateMotor(OUT_C,speed,(a3-
a3_before)/4);

ResetTachoCount(OUT_C);

RotateMotor(OUT_C,speed,(a3-
a3_before)/4);

ResetTachoCount(OUT_C);

RotateMotor(OUT_C,speed,(a3-
a3_before)/4);

ResetTachoCount(OUT_C);

flag_mAmoved=1;

flag_mCmoved=1;

a2_before=a2;

a3_before=a3;

fspeed_before=fspeed;

Wait(SEC_1);

ResetTachoCount(OUT_A);

ResetTachoCount(OUT_C);

PlayTone(500,500); // Positionnement fini

new=0;

}

if(a3!=a3_before||a2!=a2_before||fspeed!=
fspeed_before){

j=10;

new=1; //nouveaux parametres détectés

buteeC=100;

buteeB=100;

buteeA=100;

butee2B=200;

butee2A=200;

butee2C=200;

for (k=0;k<40;k++){ //retour a la
position de base A2=50;A4=38;
A6=152;

if((butee2A>5)||(butee2A<5)){

RotateMotor(OUT_A,80, 20);

if(k%2==0){détecte la position du
moteur 1 oscillation sur 2.

butee2A=MotorTachoCount(OUT_A);//
butee2A=0 ou proche de 0 si le moteur est
en bûté.

}

}

if((butee2B>20)||(butee2B<-20)){

//RotateMotor(OUT_B,60, -10);

if(k%2==0){

butee2B=MotorTachoCount(OUT_B);

}

}

if((butee2C>5)||(butee2C<-5)){

// RotateMotor(OUT_C,70, 10);

if(k%2==0){

butee2C=MotorTachoCount(OUT_C);

}

}

ResetTachoCount(OUT_C);

ResetTachoCount(OUT_B);

ResetTachoCount(OUT_A);

}

Wait(SEC_1);

ResetSleepTimer();

}

```

nxTOF_SA.nxc

```
//
// Spectrometer TOF Standalone with the
// LEGO Mindstorms NXT
//
// 2010, Institut Laue Langevin.
```

```
task main ()
```

```
{
```

```
int i, j;
```

```
int count_m = 1; // Identifiant des
moteurs 1 pour le moteur A, 2 pour le
moteur B, et 3 pour le C
```

```
int temp1=0; // 1 si nous sommes dans
un des moteur
```

```
int menuM=0; // 1 si nous sommes
dans le selecteur d'angle du
Monochromateur
```

```
int menuF=0; // 1 si nous sommes
dans le selecteur de vitesse du Fermi
```

```
int menuS=0; // 1 si nous sommes
dans le selecteur d'angle du sample
```

```
float angleM = 0; // angle du
monochromateur
```

```
float angleM_before = 0;
```

```
float speedF = 25; // vitesse de rotation
du Fermi chopper en pourcent
```

```
float angleS = 0; // angle du sample
```

```
float angleS_before = 0;
```

```
//flag permettant de stoper des boucles
```

```
int flag1=0;
```

```
int flag11=0;
```

```
int flag111=0;
```

```
int flag1111=0;
```

```
int flag2=0;
```

```
int flag3=0;
```

```
int flag33=0;
```

```
int flag333=0;
```

```
int flag4=0;
```

```
int flagsortie=0;
```

```
int flagsortie2=0;
```

```
int flagsortieM=0;
```

```
int flagsortieM2=0;
```

```
int motor_a_now, motor_a_init, a;
```

```
int motor_b_now, motor_b_init, b;
```

```
int motor_c_now, motor_c_init, c;
```

```
// positions initiales
```

```
motor_a_init = MotorTachoCount
(OUT_A);
```

```
motor_a_now = motor_a_init;
```

```
motor_b_init = MotorTachoCount
(OUT_B);
```

```
motor_b_now = motor_b_init;
```

```
motor_c_init = MotorTachoCount
(OUT_C);
```

```
motor_c_now = motor_c_init;
```

```
SetSensorTouch(IN_1); //Capteur de
pression G branché sur le capteur 1
```

```
SetSensorTouch(IN_2); //Capteur de
pression D branché sur le capteur 2
```

```
while (true)
```

```
{
```

```
if(temp1==0){
```

```
TextOut(10, LCD_LINE1, "IN6-TOF
MODEL");
```

```
TextOut(40, LCD_LINE2, "ILL");
```

```
TextOut(0, LCD_LINE4, "SELECT
MOTOR :");
```

```
TextOut(0, LCD_LINE7, "(G to change
and");
```

```
TextOut(10, LCD_LINE8, "D to
validate)");
```

```
}
```

```
if (count_m==1 && temp1==0)
```

```
{
```

```
NumOut(90, LCD_LINE4, 1);
```

```
TextOut(0, LCD_LINE5, "
");
```

```
TextOut(10, LCD_LINE5, "MONOCHROMATOR");
```

```
}
```

```
if (count_m==2 && temp1==0)
```

```
{
```

```
NumOut(90, LCD_LINE4, 2);
```

```
TextOut(0, LCD_LINE5, "
");
```

```
TextOut(10, LCD_LINE5, "FERMI
CHOPPER");
```

```
}
```

```
if (count_m==3 && temp1==0)
```

```
{
```

```
NumOut(90, LCD_LINE4, 3);
```

```
TextOut(0, LCD_LINE5, "
");
```

```

");
TextOut(30, LCD_LINE5, "SAMPLE");
}

if(SENSOR_1 == 0){
    flag1=0;
    flag2=0;
    flag11=0;
    flag111=0;
    flag1111=0;
}

if(SENSOR_1 == 1 && temp1==0 &&
flag2==0) //Sensor G activé
{

    count_m=count_m+1;
    if (count_m>3){ //limité à 3 moteurs
        count_m=1;
    }
    NumOut(90, LCD_LINE4, count_m);

    flag2=1;
}

if(SENSOR_2 == 0){
    flag3=0;
    flag33=0;
    flag333=0;
    flag4=0;
}

if(SENSOR_2 == 1 && count_m==1 &&
flag4==0) //Sensor D activé et choix du
monochromateur
{
    temp1=1;
    menuM=1;

    TextOut(0, LCD_LINE4, "
"); // efface la ligne de l'écran
    TextOut(0, LCD_LINE5, "
");
    TextOut(0, LCD_LINE6, "
");
    TextOut(0, LCD_LINE7, "
");
    TextOut(0, LCD_LINE8, "
");

    TextOut(0, LCD_LINE4,
"MONOCHROMATOR");
    TextOut(30, LCD_LINE5, "ANGLE:");
    TextOut(10, LCD_LINE7, "(G - or D
+)");
    TextOut(10, LCD_LINE8, "(G & D
finish)");

    if(SENSOR_1 ==1 && flag1==0){
        // G+D sortie du menu

        TextOut(0, LCD_LINE4, "
"); // efface la ligne de l'écran
        TextOut(0, LCD_LINE5, "
");
        TextOut(0, LCD_LINE6, "
");
        TextOut(0, LCD_LINE7, "
");
        TextOut(0, LCD_LINE8, "
");
        temp1=0;
    }

    flag1=1;
    menuM=0;
}

if(SENSOR_1 == 1)
    flag1=1;
}

////////// Definition de l'angle lié au
monochromateur //////////

if(menuM==1)
{
    if(SENSOR_2 == 1)
    {
        angleM=angleM + 0.03;
        TextOut(73, LCD_LINE5, "
");
        NumOut(73, LCD_LINE5, angleM);

        if(SENSOR_1 ==1 && flag11==0){
            TextOut(0, LCD_LINE5, "
");

            temp1=0;

            flag11=1;
            menuM=0;
        }
    }

    if(SENSOR_1 == 1)

```

<pre>{ angleM=angleM - 0.03; TextOut(73, LCD_LINE5, " "); NumOut(73, LCD_LINE5, angleM); if(SENSOR_2 ==1 && flag3==0){ TextOut(0, LCD_LINE5, " "); templ=0; flag3=1; menuM=0; } } if(SENSOR_1 == 1 SENSOR_2 == 1){ flagsortieM=0; } } if(SENSOR_1 == 0 && SENSOR_2 == 0 && flagsortieM == 0){ if((angleM-angleM_be fore)>20 (angleM-angleM_be fore)<20){ Rotate Motor(OUT_A,80,-(angleM- angleM_be fore)/4); ResetTachoCount(OUT_A); Wait(500); Rotate Motor(OUT_A,80,-(angleM- angleM_be fore)/4); ResetTachoCount(OUT_A); Wait(500);</pre>	<pre>Rotate Motor(OUT_A,80,-(angleM- angleM_be fore)/4); ResetTachoCount(OUT_A); Wait(500); Rotate Motor(OUT_A,80,-(angleM- angleM_be fore)/4); } else { Rotate Motor(OUT_A,80,-(angleM- angleM_be fore)); } ResetTachoCount(OUT_A); flagsortieM=1; flagsortieM2=0; } if (flagsortieM==1&&flagsortieM2==0){ angleM_be fore=angleM; flagsortieM2=1; } } // //// if(SENSOR_2 == 1 && count_m==2 && flag4==0) // Fèrmi chopper selectionné { templ=1; menuF=1;</pre>	<pre>TextOut(0, LCD_LINE4, " "); TextOut(0, LCD_LINE5, " "); TextOut(0, LCD_LINE6, " "); TextOut(0, LCD_LINE7, " "); TextOut(0, LCD_LINE8, " "); TextOut(0, LCD_LINE4, "FERMI CHOPPER"); TextOut(30, LCD_LINE5, "SPEED :"); TextOut(10, LCD_LINE7, "(G - or D +)"); TextOut(10, LCD_LINE8, "(G & D finish)"); if(SENSOR_1 ==1 && flag1==0){ // G+D retour au menu TextOut(0, LCD_LINE4, " "); TextOut(0, LCD_LINE5, " "); TextOut(0, LCD_LINE6, " "); TextOut(0, LCD_LINE7, " "); TextOut(0, LCD_LINE8, " "); templ=0; flag1=1; } flag4=1; } ////////// Définition de la vitesse de rotation du Fèrmi Chopper //////////</pre>
--	--	--

```

        ");
    if(menuF==1)
    {
        if(SENSOR_2 == 1)
        {
            speedF=speedF + 0.03;

            if(speedF>=100){
                speedF=100;
            }
            TextOut(73, LCD_LINE5, "
");
            NumOut(73, LCD_LINE5, speedF);

            if(SENSOR_1 ==1 && flag111==0){

                TextOut(0, LCD_LINE5, "
");

                temp1=0;
                flag111=1;
                menuF=0;

            }
        }

        if(SENSOR_1 == 1)
        {
            speedF=speedF - 0.03;
            if(speedF<=0){
                speedF=0;
            }

            TextOut(73, LCD_LINE5, "
");
            NumOut(73, LCD_LINE5, speedF);
            if(SENSOR_2 ==1 && flag33==0){

                TextOut(0, LCD_LINE5, "

        ");
        if(SENSOR_1 ==1 && flag1=1){
            // G+D retour au menu principal
            TextOut(0, LCD_LINE4, "
");
            TextOut(0, LCD_LINE5, "
");
            TextOut(0, LCD_LINE6, "
");
            TextOut(0, LCD_LINE7, "
");
            TextOut(0, LCD_LINE8, "
");
            temp1=0;
            flag1=1;
        }
        flag4=1;
    }

    if(SENSOR_2 == 1 && count_m==3 &&
    flag4==0) // Selection du menu lié au
    sample
    {
        temp1=1;
        menuS=1;

        TextOut(0, LCD_LINE4, "
");
        TextOut(0, LCD_LINE5, "
");
        TextOut(0, LCD_LINE6, "
");
        TextOut(0, LCD_LINE7, "
");
        TextOut(0, LCD_LINE8, "
");

        TextOut(30, LCD_LINE4, "SAMPLE");
        TextOut(30, LCD_LINE5, "ANGLE:");
        TextOut(10, LCD_LINE7, "(G - or D
+)"");
        TextOut(10, LCD_LINE8, "(G & D
finish)");

        if(menuS==1)
        {
            if(SENSOR_2 == 1)
            {
                angleS=angleS + 0.03;

                TextOut(73, LCD_LINE5, "
");
                NumOut(73, LCD_LINE5, angleS);
                if(SENSOR_1 ==1 && flag11==0){

```

```

TextOut(0, LCD_LINE5, "
");
temp1=0;
flag1=1;
menuS=0;
}
}

if(SENSOR_1 == 1)
{
angleS=angleS - 0.03;
TextOut(73, LCD_LINE5, "
");
NumOut(73, LCD_LINE5, angleS);
if(SENSOR_2 ==1 && flag3==0){
TextOut(0, LCD_LINE5, "
");

temp1=0;
flag3=1;
menuS=0;
}
}

if(SENSOR_1 == 1 || SENSOR_2 == 1){
flagsortie=0;
}

if(SENSOR_1 == 0 && SENSOR_2 == 0
&& flagsortie == 0){

RotateMotor(OUT_C,60,angleS/2-
angleS_before/2);

ResetTachoCount(OUT_C);

flagsortie=1;

flagsortie2=0;
}

if (flagsortie==1 && flagsortie2==0){
angleS_before=angleS;
flagsortie2=1;
}

////////////////////////////////////
////
}
}

```


nxTAS_SA

```
//
// Spectrometer TAS Standalone with the
// LEGO Mindstorms NXT
//
// 2010, Institut Laue Langevin.
```

```
task main ()
```

```
{
int i=0;
int count_m = 1; // Identifiant des
moteurs 1 pour le moteur A, 2 pour le
moteur B, et 3 pour le C
```

```
int temp1=0; // 1 si nous sommes dans
un des menus moteur
int temp1A=0;
int menuM=0; // 1 si nous sommes
dans le selecteur d'angle du
Monochromateur
int menuA=0; // 1 si nous sommes
dans le selecteur d'angle de l'analyseur
int menuS=0; // 1 si nous sommes
dans le selecteur d'angle du sample
```

```
float positionM=0; //somme des angles
pour le monochromateur rentrés depuis
le lancement du programme
float positionS=0; //somme des angles
pour le sample rentrés depuis le
lancement du programme
float positionA=0; //somme des angles
pour l'analyseur rentrés depuis le
lancement du programme
```

```
float angleM=0; // Monochromator
angle
float angleM_before=0;
float angleA=0; // Analyzer angle
float angleA_before=0;
float angleS=0; //Sample angle
float angleS_before=0;
```

```
float countmotor=0;
```

```
float countmotorM=0;
```

```
//flag permettant de stoper des boucles
int flag1=0;
int flag11=0;
int flag111=0;
int flag111A=0;
int flag2=0;
```

```
int flag1A=0;
int flag11A=0;
int flag111A=0;
int flag1111A=0;
```

```
int flag3=0;
int flag33=0;
int flag33A=0;
```

```
int flagA=0;
int flag333=0;
int flag4=0;
int flag4A=0;
```

```
int flagsortie=0;
int flagsortie2=0;
```

```
int flagsortieM=0;
int flagsortieM2=0;
```

```
int flagsortieA=0;
int flagsortieA2=0;
```

```
int flag_positionM=0;
int flag_positionS=0;
int flag_positionA=0;
```

```
int flag_amort=0;
int flag_amort2=0;
```

```
int motor_a_now, motor_a_init, a;
int motor_b_now, motor_b_init, b;
int motor_c_now, motor_c_init, c;
```

```
// positions initiales
motor_a_init = MotorTachoCount
(OUT_A);
motor_a_now = motor_a_init;
```

```
motor_b_init = MotorTachoCount
(OUT_B);
motor_b_now = motor_b_init;
```

```
motor_c_init = MotorTachoCount
(OUT_C);
motor_c_now = motor_c_init;
```

```
SetSensorTouch(IN_1); //Capteur de
pression G branché sur le capteur 1
SetSensorTouch(IN_2); //Capteur de
pression D branché sur le capteur 2
```

```
while (true)
{
```

```
NumOut(20, LCD_LINE6,
MotorTachoCount(OUT_A));
NumOut(50, LCD_LINE6,
flag_positionS);
```

```
if((MotorTachoCount(OUT_B)!=0)) &&
flag_amort==0){ //asservissement
```

```
NumOut(80, LCD_LINE6, 1);
```

```
for(i=1;i<3;i++){
countmotor = -
MotorTachoCount(OUT_B);
RotateMotor(OUT_B,90, countmotor);
}
```

```
flag_amort=1;
```

```
ResetTachoCount(OUT_B);
} else {
NumOut(80, LCD_LINE6, 0);
}
```

```
if(((MotorTachoCount(OUT_A)!=0)) &&
flag_amort2==1){ //asservissement du
sample pour éviter qu'il ne bouge en
bougeant l'analyser
```

```
NumOut(90, LCD_LINE6, 1);
```

```
for(i=1;i<3;i++){
countmotorM = -
MotorTachoCount(OUT_A);
RotateMotor(OUT_A,90, countmotorM);
}
```

```
flag_amort2=0;
ResetTachoCount(OUT_A);
} else {
NumOut(90, LCD_LINE6, 0);
}
```

```
if(temp1==0){
TextOut(10, LCD_LINE1, "IN6-TAS
MODEL");
TextOut(40, LCD_LINE2, "ILL");
TextOut(0, LCD_LINE4, "SELECT
MOTOR :");
TextOut(0, LCD_LINE7, "(G to change
and)");
TextOut(10, LCD_LINE8, "D to
validate)");
}
```

```
if (count_m==1 && temp1==0)
{
NumOut(90, LCD_LINE4, 1);
TextOut(0, LCD_LINE5, "
");
TextOut(10, LCD_LINE5,
"MONOCHROMATOR");
}
```

```
if (count_m==2 && temp1==0)
{
NumOut(90, LCD_LINE4, 2);
TextOut(0, LCD_LINE5, "
");
TextOut(30, LCD_LINE5, "SAMPLE");
}
```

```
if (count_m==3 && temp1==0)
{
NumOut(90, LCD_LINE4, 3);
TextOut(0, LCD_LINE5, "
");
TextOut(30, LCD_LINE5,
"ANALYZER");
}
```

```

TextOut(10, LCD_LINE7, "(G - or D
+");
TextOut(10, LCD_LINE8, "(G & D
finish)");

if(SENSOR_1 == 0){
flag1=0;
flag2=0;
flag11=0;
flag111=0;
flag1A=0;
flag11A=0;
flag111A=0;
flag111A=0;
}

if(SENSOR_2 == 0){
flag3=0;
flag33=0;
flag33A=0;
flagA=0;
flag333=0;
flag4=0;
flag4A=0;
}

if(SENSOR_1 == 1 && temp1==0 &&
flag2==0) // capteur G activé
{
count_m=count_m+1;
if (count_m>3){ // il y a 3 moteurs
count_m=1;
}
NumOut(90, LCD_LINE4, count_m);

flag2=1;
}

if(SENSOR_2 == 1 && count_m==1 &&
flag4==0) // Menu monochromateur
{
temp1=1;
menuM=1;

TextOut(0, LCD_LINE4, "
"); //efface les lignes de l'ecran
TextOut(0, LCD_LINE5, "
");
TextOut(0, LCD_LINE6, "
");
TextOut(0, LCD_LINE7, "
");
TextOut(0, LCD_LINE8, "
");

TextOut(0, LCD_LINE4,
"MONOCHROMATOR");
TextOut(30, LCD_LINE5, "ANGLE :");

TextOut(10, LCD_LINE7, "(G - or D
+");
TextOut(10, LCD_LINE8, "(G & D
finish)");

if(SENSOR_1 == 1 && flag1==0){
// Go back to menu G+D
TextOut(0, LCD_LINE4, "
"); //Erase screen datas
TextOut(0, LCD_LINE5, "
");
TextOut(0, LCD_LINE6, "
");
TextOut(0, LCD_LINE7, "
");
TextOut(0, LCD_LINE8, "
");
temp1=0;

flag1=1;
}

flag4=1;
}

////////// Define angle of the
Monochromator //////////

if(menuM==1)
{
flag_amort=0; // Analyseur asservi si ca
bouge
flag_positionM=1;
if(SENSOR_2 == 1)
{
angleM=angleM + 0.06;
TextOut(73, LCD_LINE5, "
");
NumOut(73, LCD_LINE5, angleM/2);

if(SENSOR_1 == 1 && flag11==0){
//efface les lignes de l'ecran
TextOut(0, LCD_LINE5, "
");
temp1=0;

flag11=1;
menuM=0;
}

if(SENSOR_1 == 1)
{
angleM=angleM - 0.06;

TextOut(73, LCD_LINE5, "
");
NumOut(73, LCD_LINE5, angleM/2);

if(SENSOR_2 == 1 && flag3==0){
//efface les lignes de l'ecran
TextOut(0, LCD_LINE5, "
");

temp1=0;

flag3=1;
menuM=0;
}

if(SENSOR_1 == 1 || SENSOR_2 == 1){
flagsortieM=0;
}

if(SENSOR_1 == 0 && SENSOR_2 == 0
&& flagsortieM == 0){

if((angleM/2-angleM_before/2)>20 ||
(angleM/2-angleM_before/2)<-20 ){
Rotate Motor(OUT_A,100,(angleM/2-
angleM_before/2)/4);
ResetTachoCount(OUT_A);
Wait(10);
Rotate Motor(OUT_A,100,(angleM/2-
angleM_before/2)/4);
ResetTachoCount(OUT_A);
Wait(10);
Rotate Motor(OUT_A,100,(angleM/2-
angleM_before/2)/4);
ResetTachoCount(OUT_A);
Wait(10);
Rotate Motor(OUT_A,100,(angleM/2-
angleM_before/2)/4);
}
else{
Rotate Motor(OUT_A,100,(angleM/2-
angleM_before/2));
}

positionM = positionM + (angleM/2-
angleM_before/2);
ResetTachoCount(OUT_A);

flagsortieM=1;
flagsortieM2=0;
}

if (flagsortieM==1 && flagsortieM2==0){
angleM_before=angleM;
flagsortieM2=1;
}

}

//////////

```

```

TextOut(73, LCD_LINE5, "
");
NumOut(73, LCD_LINE5, angleS*2);

if(SENSOR_2 == 1 && count_m==2 &&
flag4==0 && flag_positionM==1) //
menu du Sample
{

flag_positionS=1;
temp1=1;
menuS=1;

TextOut(0, LCD_LINE4, "
");
TextOut(0, LCD_LINE5, "
");
TextOut(0, LCD_LINE6, "
");
TextOut(0, LCD_LINE7, "
");
TextOut(0, LCD_LINE8, "
");

TextOut(30, LCD_LINE4, "SAMPLE");
TextOut(30, LCD_LINE5, "ANGLE:");
TextOut(10, LCD_LINE7, "(G - or D
+)");
TextOut(10, LCD_LINE8, "(G & D
finish)");

if(SENSOR_1 == 1 && flag1=1){
// G+D retour au menu principal
TextOut(0, LCD_LINE4, "
");
TextOut(0, LCD_LINE5, "
");
TextOut(0, LCD_LINE6, "
");
TextOut(0, LCD_LINE7, "
");
TextOut(0, LCD_LINE8, "
");
temp1=0;
flag1=1;
}
flag4=1;
}

////////// Définition de l'angle du
sample //////////

if(menuS==1)
{
flag_amort2=1;

if(SENSOR_2 == 1)
{
angleS=angleS + 0.015;

TextOut(73, LCD_LINE5, "
");
NumOut(73, LCD_LINE5, angleS*2);

if(SENSOR_1 == 1 && flag1==0){
TextOut(0, LCD_LINE5, "
");
flag1=1;
menuS=0;
temp1=0;
}

if(SENSOR_1 == 1)
{
angleS=angleS - 0.015;

TextOut(73, LCD_LINE5, "
");
NumOut(73, LCD_LINE5, angleS*2);

if(SENSOR_2 == 1 && flag3==0){
TextOut(0, LCD_LINE5, "
");
temp1=0;
flag3=1;
menuS=0;
}

if(SENSOR_1 == 1 || SENSOR_2 == 1){
flagsortie=0;
}

if(SENSOR_1 == 0 && SENSOR_2 == 0
&& flagsortie == 0){

if((angleS-angleS_before)>20 || (angleS-
angleS_before)<-20){
RotateMotor(OUT_B,100,(angleS-
angleS_before)/4);
ResetTachoCount(OUT_B);
Wait(10);
RotateMotor(OUT_B,100,(angleS-
angleS_before)/4);
ResetTachoCount(OUT_B);
Wait(10);
RotateMotor(OUT_B,100,(angleS-
angleS_before)/4);
ResetTachoCount(OUT_B);
Wait(10);
RotateMotor(OUT_B,100,(angleS-
angleS_before)/4);
}
else{
RotateMotor(OUT_B,100,(angleS-
angleS_before));
}

positionS = positionS + (angleS-
angleS_before);
ResetTachoCount(OUT_B);

flagsortie=1;
flagsortie2=0;
}

if (flagsortie==1&&flagsortie2==0){
angleS_before=angleS;
flagsortie2=1;
}

}

//////////
///

if(SENSOR_2 == 1 && count_m==3 &&
flag4==0 && flag_positionS==1) // menu
de l'analyseur
{
temp1=1;
menuA=1;

TextOut(0, LCD_LINE4, "
");
TextOut(0, LCD_LINE5, "
");
TextOut(0, LCD_LINE6, "
");
TextOut(0, LCD_LINE7, "
");
TextOut(0, LCD_LINE8, "
");

TextOut(0, LCD_LINE4,
"ANALYZER");
TextOut(30, LCD_LINE5, "ANGLE:");
TextOut(10, LCD_LINE7, "(G - or D
+)");
TextOut(10, LCD_LINE8, "(G & D
finish)");

if(SENSOR_1 == 1 && flag1==0){
// G+D retour au menu principal
TextOut(0, LCD_LINE4, "
");
TextOut(0, LCD_LINE5, "
");
TextOut(0, LCD_LINE6, "
");
TextOut(0, LCD_LINE7, "
");
TextOut(0, LCD_LINE8, "
");
}

```

```
temp1=0;
flag1=1;

}
flag4=1;
}
```

```
////////// Définition de l'angle de
l'analyseur //////////
```

```
if(menuA==1)
{
flag_positionA=1;
}
```

```
if(SENSOR_2 == 1)
{
```

```
angleA=angle A + 0.03;
```

```
TextOut(73, LCD_LINE5, "
");
NumOut(73, LCD_LINE5, angle A);
```

```
if(SENSOR_1 == 1 && flag11A==0){
```

```
TextOut(0, LCD_LINE5, "
```

```
");
```

```
temp1=0;
flag11A=1;
menuA=0;
}
```

```
}
```

```
if(SENSOR_1 == 1)
{
```

```
angleA=angle A - 0.03;
```

```
TextOut(73, LCD_LINE5, "
");
NumOut(73, LCD_LINE5, angle A);
```

```
if(SENSOR_2 == 1 && flagA==0){
```

```
TextOut(0, LCD_LINE5, "
");
```

```
temp1=0;
flagA=1;
menuA=0;
}
```

```
}
```

```
if(SENSOR_1 == 1 || SENSOR_2 == 1){
flagsortieA=0;
}
```

```
if(SENSOR_1 == 0 && SENSOR_2 == 0
&& flagsortieA == 0){
```

```
Rotate Motor(OUT_C,100,angleA-
angleA_before);
```

```
ResetTachoCount(OUT_C);
```

```
flagsortieA=1;
flagsortieA2=0;
}
```

```
if (flagsortieA==1&&flagsortieA2==0){
angleA_before=angleA;
flagsortieA2=1;
}
```

```
}
```

```
////////////////////////////////////////
//////
```

```
}
```

```
}
```

Annexe 5 : Documents du package.

installer NXT ubuntu – transferer les programmes.doc

Installer NXT et NXC sur Ubuntu 10.04

(basé sur http://doc.ubuntu-fr.org/lego_mindstorms_nxt_sur_ubuntu)

fonctionnel au 10/11/2010

Avant-propos :

Ce document permet à l'utilisateur d'installer le compilateur NBC qui gère les fichiers .nxc, programmes de contrôle du NXT écrits en NXC (Not Exactly C) et de configurer les paramètres de communication entre la brique NXT et Ubuntu.

1/ Téléchargement

NBC Beta Releases(1.2.1 r3) : <http://downloads.sourceforge.net/bricxcc/nbc-1.2.1.r3.src.tgz>

Talk 2 NXT : <http://www-verimag.imag.fr/~raymond/edu/lego/t2n/t2n-0.2.tgz>

Extraction et placement du contenu dans **/usr/bin**. Ouvrir un terminal et taper :

```
$ cd Desktop
$ tar xzf nbc-1.2.1.r3.tgz
$ tar xzf t2n-0.2.tgz
$ sudo cp 'NXT/nbc' '/usr/bin'
$ sudo cp 't2n-0.2/t2n' '/usr/bin'
```

2/ Compilation

Pour tester que tout fonctionne bien, compilons **nxTAS.nxc**.

```
$ ls
nxTAS.nxc
$ nbc -EF nxTAS.nxc -O=nxTAS.rxe
$ ls
nxTAS.rxe nxTAS.nxc
```

nxTAS.rxe est notre programme compilé et peut-être utilisé comme tel par le NXT.

3/ Communication

Connectez via USB le NXT à l'ordinateur et le mettre sous tension.

Entrez la commande suivante :

\$ lsusb

L'ensemble des périphériques USB connectés est notifié et une ligne doit indiquer que le NXT est correctement reconnu par le système. Cette ligne doit ressembler à celle-ci :

Bus 004 Device 007: ID 0694:0002 Lego Group

Entrez alors la commande suivante :

```
$ t2n -i
```

* Si cette commande génère une erreur, il faut copier manuellement le fichier **70-lego.rules** présent dans le dossier '**t2n-0.2/udev/**' dans '**/etc/udev/rules.d/**'. Puis redémarrez.

La commande t2n -i, si elle fonctionne, génère un rapport de type :

```
#### NXT INFOS #####  
protocol version=1.124 firmware version=1.1  
NXT Name: NXT  
Bluetooth address: XX:XX:XX:XX:XX:XX  
Bluetooth signal: 0  
Free user flash: 57116
```

Testez la communication en transférant le fichier précédemment compilé nxTAS.rxe.

```
$ ls  
nxTAS.rxe  
$ t2n -put nxTAS.rxe
```

Un son aigue provenant du NXT signale que le transfert s'est achevé et que le programme est prêt à être utilisé.

NBC, le compilateur de fichiers NXC permet également de transférer les fichiers sans passer par l'étape "t2n". Ainsi pour compiler, puis transférer un programme en une seule ligne de commande, préférez :

```
$ nbc -EF -d nxTAS.nxc
```

4/ Lancer le programme

Allumez le NXT puis allez dans My Files>Software Files. Vous y trouverez les programmes transférés.

Modèles CAO –liste pieces.doc



Modèles CAO du nxTAS et nxTOF

fonctionnel au 10/11/2010

Avant-propos :

Télécharger le logiciel Lego Digital Designer et suivre ce document permettront à l'utilisateur d'avoir accès au manuel de construction animé, bien plus clair et ludique que le manuel PDF fourni en fichier joint. Il permettra également de modifier le modèle proposé et d'en obtenir une liste des pièces avec leurs références LEGO en vu d'une commande.

1/ Téléchargement

Lego Digital Designer (LDD v4.0)

Windows : http://cache.lego.com/downloads/ldd2.0/installer/SetupLDD-PC-4_0_20.exe

Mac : http://cache.lego.com/downloads/ldd2.0/installer/SetupLDD-MAC-4_0_20.zip

Pour générer la liste des pièces uniquement : **MLCad (v3.30)**

Windows : http://www.lm-software.com/mlcad/MLCad_V3.30.zip

Linux : <http://www.lm-software.com/mlcad/Linux.htm>

Fichiers joints

liste TAS.txt : liste des pièces à commander pour le modèle TAS.

liste TOF.txt : liste des pièces à commander pour le modèle TOF.

TAScommande.lxf : Modèle CAO du TAS sans les pièces NXT (moteurs, brique, capteurs) mais avec quelques pièces de construction supplémentaires compatible avec Lego Digital Designer.

TASwith.lxf : Modèle CAO du TAS avec les pièces NXT (moteurs, brique, capteurs) compatible avec Lego Digital Designer.

TOFcommande.lxf : Modèle CAO du TOF sans les pièces NXT (moteurs, brique, capteurs) mais avec quelques pièces de construction supplémentaires compatible avec Lego Digital Designer.

TOFwith.lxf : Modèle CAO du TAS avec les pièces NXT (moteurs, brique, capteurs) compatible avec Lego Digital Designer.

TOFTAScommande.lxf : Modèle CAO du TOF et TAS sans les pièces NXT (moteurs, brique, capteurs) mais avec quelques pièces de construction supplémentaires compatible avec Lego Digital Designer.

TOF building instuctions.pdf : Manuel de construction du TOF.

TAS building instuctions.pdf : Manuel de construction du TAS.

2/ Manuel de construction interactif

Ouvrir l'un des deux fichiers **TASwith.lxf** ou **TOFwith.lxf** dans LDD. Passer au “Building guide mode” en cliquant sur la troisième icône en haut à droite de la fenêtre.



Le logiciel va alors générer le manuel utilisateur interactif et papier.

ATTENTION : Lire la partie **6/ Limites des modèles**

3/ Modifier les modèles

Les fichiers sont libres en écriture et lecture. Le logiciel est très simple d'utilisation, on charge et place les pièces depuis le catalogue par de simple Drag&Drop. Des tutoriaux d'utilisation sont disponibles sur la page officielle du logiciel. <http://ldd.lego.com/getstarted/default.aspx>

4/ Générer la liste des pièces

Dans **Lego Digital Designer**, exporter le modèle au format **LDraw**.
Lancer **MLCad** et ouvrir le fichier exporté au format **Ldraw**.

Allez dans Extras>Reports>Parts

Cochez l'option "Ignore colors" et sauvez cette liste.

Ouvrez le fichier *.txt. Il sera de la forme suivante.

No.	Part no.	Part name
5	3705.dat	Technic Axle 4

Il faut alors lire que la maquette contient 5 fois la pièce dont la référence LEGO est **3705** et qui se nomme "Technic Axle 4". Les modèles TOF et TAS contiennent une trentaine de références.

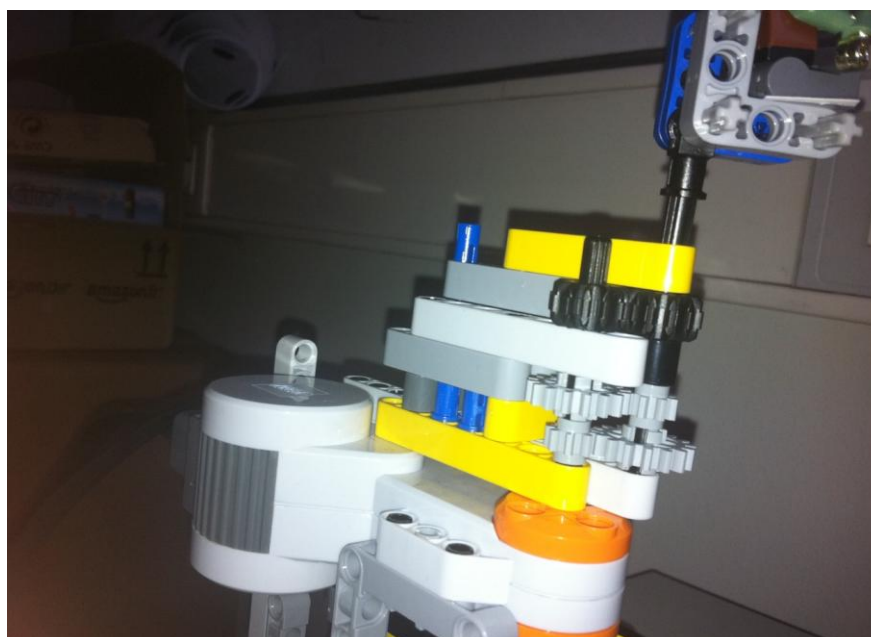
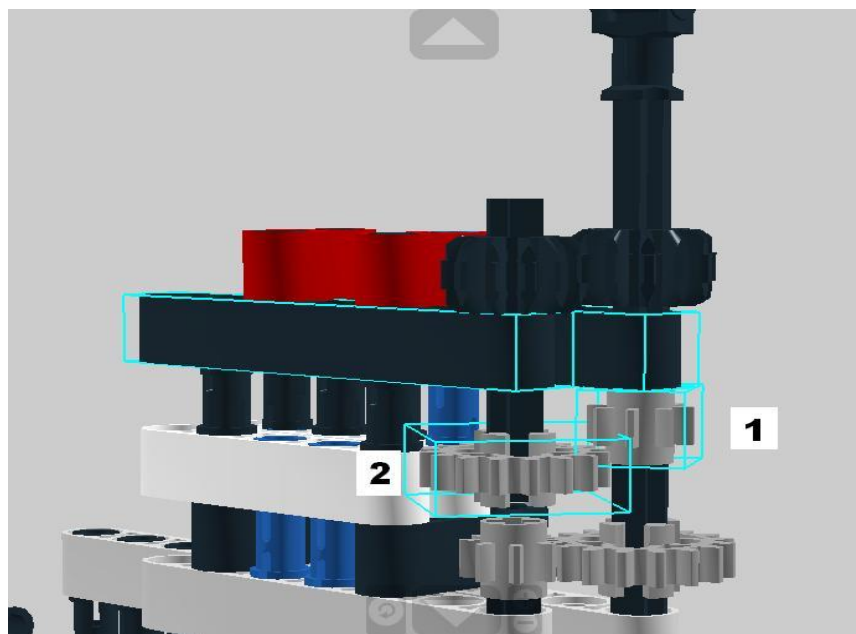
Cf. liste des pièces des 2 modèles en dernière page de ce document.

5/ Commander les pièces

Il suffit de téléphoner à LEGO et de communiquer les références des pièces. Chaque commande est limitée à une quinzaine de références. Il faudra alors sans doute faire 2 commandes pour chaque modèle.

6/ Limites des modèles

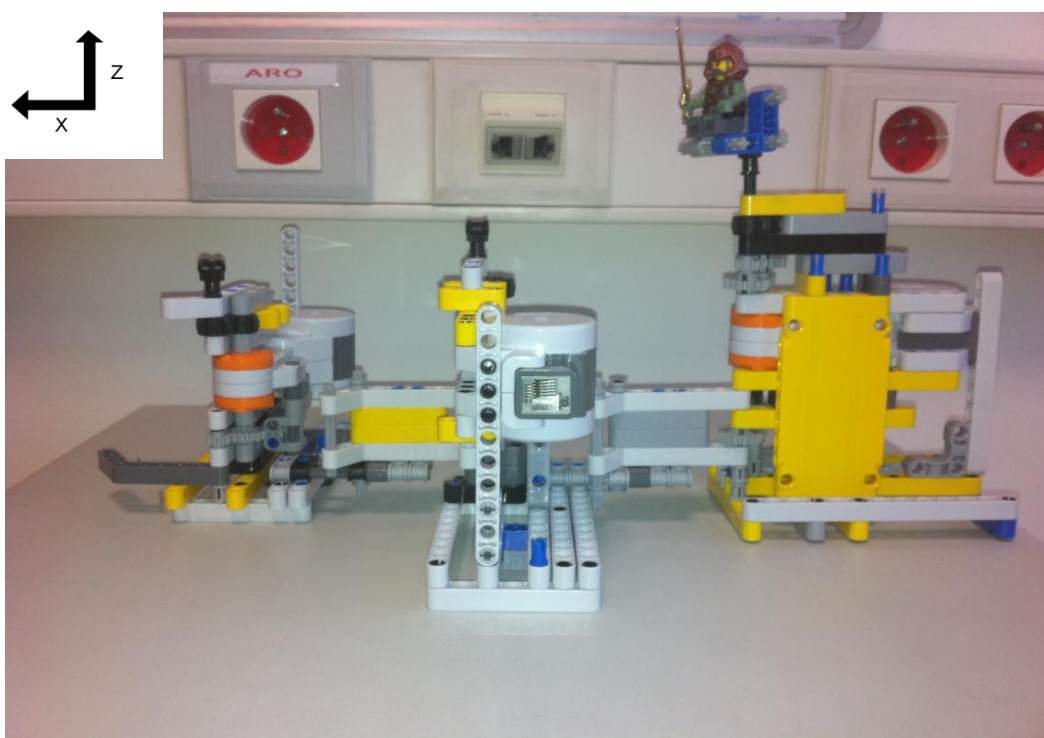
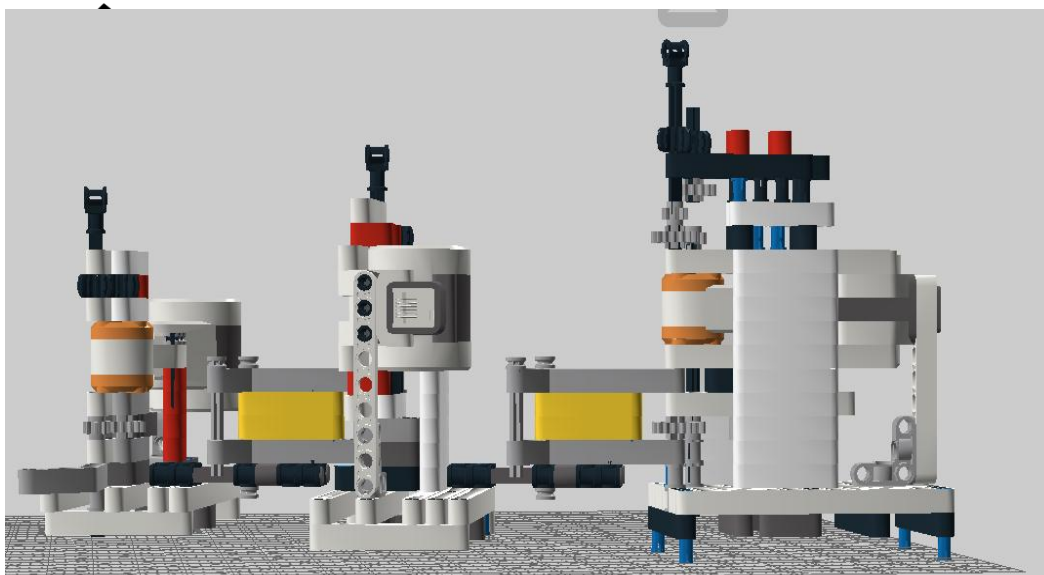
Les contraintes du logiciel ont empêché l'alignement des engrenages 1 et 2 sur la figure ci-dessous. Ces engrenages font partie de l'ensemble supportant le monochromateur. Le problème d'alignement nous a obligés à surélever le dernier "étage". Il convient donc, lors du montage final, de redescendre ces briques comme montré dans la figure du modèle réel ci-dessous.



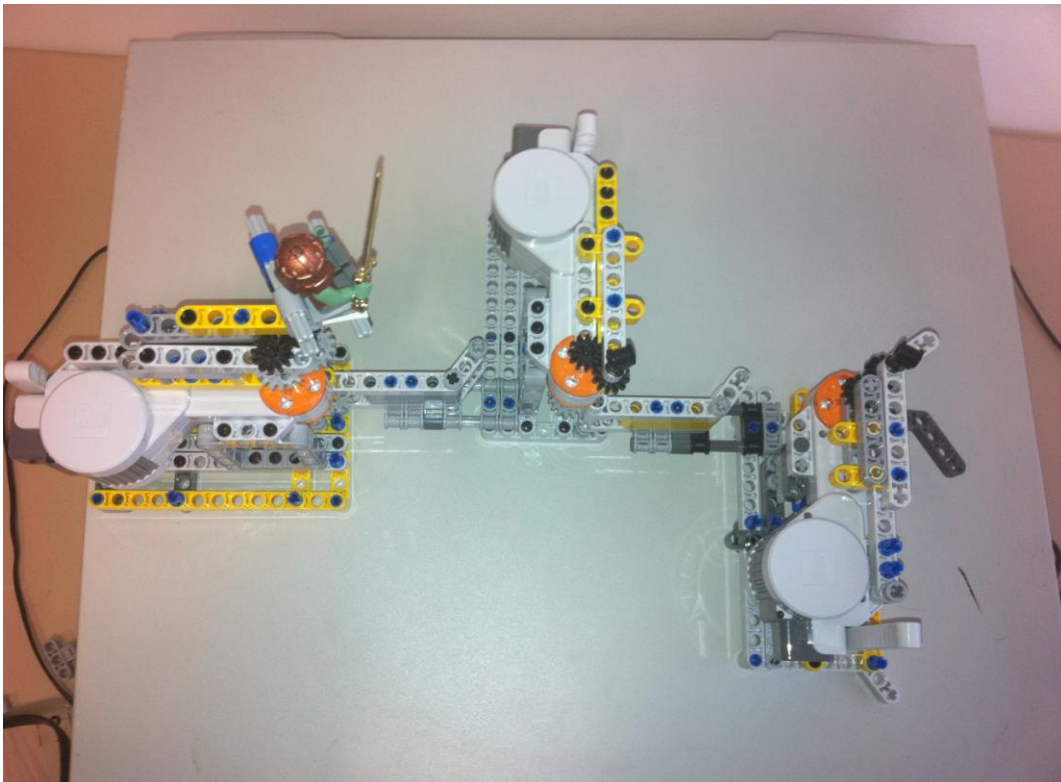
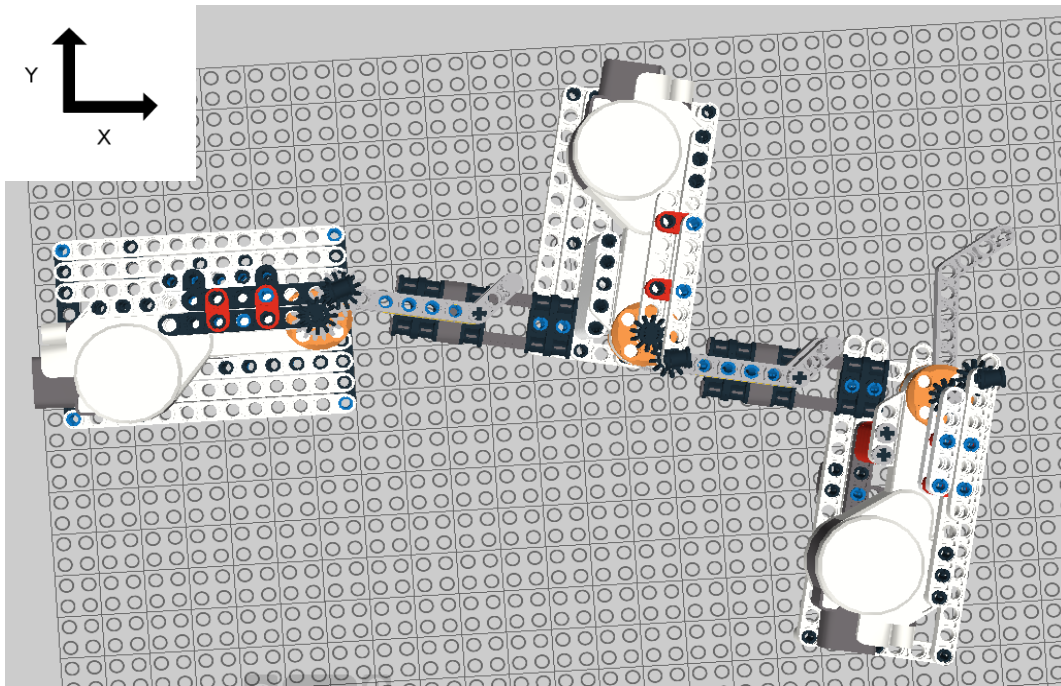
7/ Coupe du modèle TAS : CAO/réal



Vue transversale droite



Vue transversale gauche



Vue du dessus

Commande TAS

No.	Part no.	Part name
<hr/>		
5	3705.dat	Technic Axle 4
2	32073.dat	Technic Axle 5
3	3706.dat	Technic Axle 6
2	3707.dat	Technic Axle 8
4	3708.dat	Technic Axle 12
1	3749.dat	Technic Axle Pin
7	43093.dat	Technic Axle Pin with Friction
11	43857.dat	Technic Beam 2
9	32140.dat	Technic Beam 2 x 4 Liftarm Bent 90
7	32523.dat	Technic Beam 3
3	32526.dat	Technic Beam 3 x 5 Bent 90
4	32271.dat	Technic Beam 3 x 7 Liftarm Bent 53.13
1	6629.dat	Technic Beam 4 x 6 Liftarm Bent 53.13
37	32316.dat	Technic Beam 5
14	32524.dat	Technic Beam 7
7	40490.dat	Technic Beam 9
6	32525.dat	Technic Beam 11
20	41239.dat	Technic Beam 13
4	32278.dat	Technic Beam 15
10	3713.dat	Technic Bush
30	32123.dat	Technic Bush 1/2
3	32039.dat	Technic Connector with Axlehole
10	32184.dat	Technic Cross Block 1 x 3 (Axle/Pin/Axle)
10	3647.dat	Technic Gear 8 Tooth
12	32270.dat	Technic Gear 12 Tooth Double Bevel
10	4019.dat	Technic Gear 16 Tooth
59	6558.dat	Technic Pin Long with Friction and Slot
161	2780.dat	Technic Pin with Friction and Slots
3	55615.dat	Technic Beam 3 x 3 Bent with Pins

Commande TOF

No.	Part no.	Part name
-----	----------	-----------

2	6143.dat	Brick 2 x 2 Round Type 2
2	3958.dat	Plate 6 x 6
1	32062.dat	Technic Axle 2 Notched
3	3705.dat	Technic Axle 4
3	32073.dat	Technic Axle 5
1	3706.dat	Technic Axle 6
1	3707.dat	Technic Axle 8
2	3737.dat	Technic Axle 10
11	3708.dat	Technic Axle 12
2	43093.dat	Technic Axle Pin with Friction
2	43857.dat	Technic Beam 2
2	32140.dat	Technic Beam 2 x 4 Liftarm Bent 90
5	32523.dat	Technic Beam 3
2	32526.dat	Technic Beam 3 x 5 Bent 90
35	32316.dat	Technic Beam 5
4	32524.dat	Technic Beam 7
7	40490.dat	Technic Beam 9
2	32525.dat	Technic Beam 11
24	41239.dat	Technic Beam 13
60	3713.dat	Technic Bush
1	32039.dat	Technic Connector with Axlehole
8	32184.dat	Technic Cross Block 1 x 3 (Axle/Pin/Axle)
8	3647.dat	Technic Gear 8 Tooth
8	32270.dat	Technic Gear 12 Tooth Double Bevel
8	4019.dat	Technic Gear 16 Tooth
1	3673.dat	Technic Pin
43	6558.dat	Technic Pin Long with Friction and Slot
129	2780.dat	Technic Pin with Friction and Slots
4	32271.dat	Technic Beam 3 x 7 Liftarm Bent 53.5
51	32123.dat	Technic Bush 1/2
3	55615.dat	Technic Beam 3 x 3 Bent with Pins

Le coup de chaque commande est d'environ 80 euros. A ce prix, s'ajoute l'achat d'une brique NXT (150€), de 3 servomoteurs (45€), de 2 capteurs de pression (35€) et des cables connecteurs (11€). L'ensemble se porte alors à environ **400 €** (pour l'achat des 2 commandes ci-avant).

Utiliser les maquettes.doc

Utiliser les maquettes nxTAS et nxTOF

fonctionnel au 10/11/2010

Avant-propos :

Les 2 maquettes sont utilisables de 2 façons différentes. Les codes **nxTAS_SA.nxc** et **nxTOF_SA.nxc** permettent de contrôler la maquette sans ordinateur. Les paramètres sont alors rentrés par l'utilisateur qui navigue dans un menu à l'aide des 2 capteurs de pression fournis en standard dans le kit NXT v2.0. Les codes **nxTAS.nxc** et **nxTOF.nxc** sont destinés à être couplé avec des logiciels interagissant en temps réel avec eux. Il est à noter que dans la suite du document, l'interaction se fera avec **McStas** sur ubuntu, mais tout logiciel et sur n'importe quelle plateforme peut-être adapté assez facilement.

1/ Téléchargement

McStas (v1.12b) : <http://www.mcstas.org/download/>

Fichiers joints

nxTAS.nxc : Programme du TAS utilisant des paramètres envoyés automatiquement.

nxTOF.nxc : Programme du TOF utilisant des paramètres envoyés automatiquement.

nxTAS_SA.nxc : Programme du TAS utilisant les paramètres rentrés à l'aide des capteurs de pression.

nxTOF_SA.nxc : Programme du TOF utilisant les paramètres rentrés à l'aide des capteurs de pression.

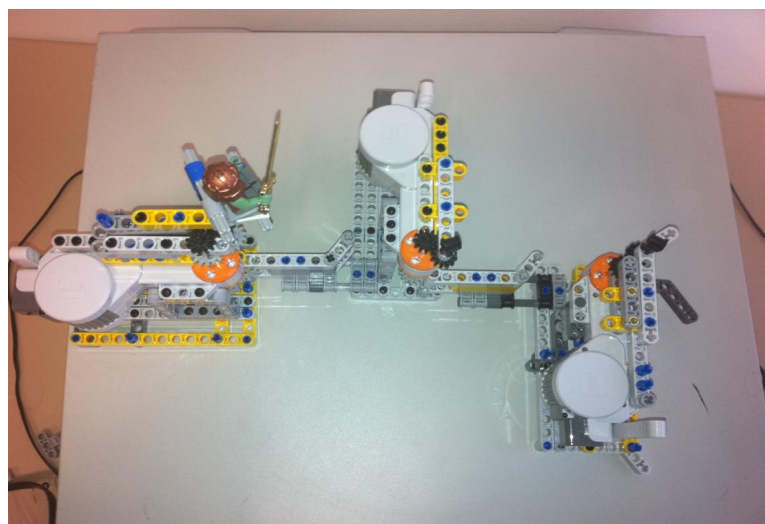
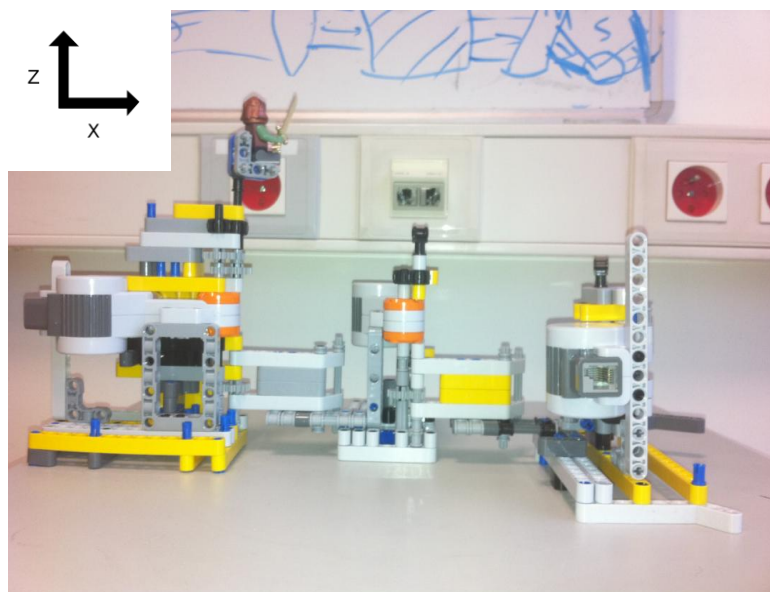
NXTAS.comp : Définition du composant nxTAS pour McStas.

nxtas.instr : Instrument faisant un simple appel du composant NXTAS.comp.

NXTOF.comp : Définition du composant nxTAS pour McStas.

nxtof.instr : Instrument faisant un simple appel du composant NXTOF.comp.

Différentes vues de la maquette TAS :



3/ A/ Utiliser la maquette en version SA (standalone)

Pré-requis :

- Télécharger les programmes **nxTAS_SA.nxc** et **nxTOF_SA.nxc** dans la brique NXT.
- Relier le port **A** au moteur du monochromateur.
- Relier le port **B** au moteur du Fermi (pour le TOF) ou du sample (pour le TAS).
- Relier le port **C** est relié au moteur du sample (pour le TOF) ou de l'analyseur (pour le TAS).
- Relier le port **1** au capteur de pression qu'on identifiera par la suite par la lettre **G**.
- Relier le port **2** au capteur de pression qu'on identifiera par la suite par la lettre **D**.
- Placer le modèle et l'aligner de manière à ce que tous les angles soient égaux à 0. (cf. figure ci-avant)

a. Lancer le programme

Le menu du programme apparaît à l'écran du NXT. Ce menu va permettre de naviguer entre les 3 sous-menus liés aux 3 moteurs et ainsi de spécifier les angles ou les vitesses de rotation.

Le bouton **G** permet de faire défiler les sous-menus :

Pour le TOF : **Monochromator**, **Fermi-Chopper** et **Sample**

Pour le TAS : **Monochromator**, **Sample** et **Analyser**

b. Entrer dans un des sous-menus

Le bouton **D** permet de rentrer dans le sous-menu sélectionné. Pour sortir du sous-menu, il suffit de presser en même temps les boutons **D** et **G**.

ATTENTION : Pour le TAS, il n'est pas possible de rentrer les paramètres du Sample ou de l'Analyser avant celui du monochromateur. Cette mesure vise à réduire les problèmes de précision.

3. Spécifier ses paramètres

Dans chaque sous-menu, utiliser **G** ou **D** pour augmenter ou diminuer la valeur de l'angle ou de la vitesse de rotation sélectionnée.

Comportement de la maquette : Une fois la valeur choisie, le moteur se positionne. Chaque moteur est asservi. En effet, les mouvements des autres moteurs vont influencer très certainement le positionnement du premier. Ainsi, pour tout mouvement détecté après un positionnement, une force contraire est appliquée.

3/ B/ Utiliser la maquette couplée avec McStas

- Télécharger les programmes **nxTAS.nxc** et **nxTOF.nxc** dans la brique NXT.
- Relier le port **A** au moteur du monochromateur.
- Relier le port **B** au moteur du Fermi (pour le TOF) ou du sample (pour le TAS).
- Relier le port **C** est relié au moteur du sample (pour le TOF) ou de l'analyseur (pour le TAS).

a/ Ouvrir McStas

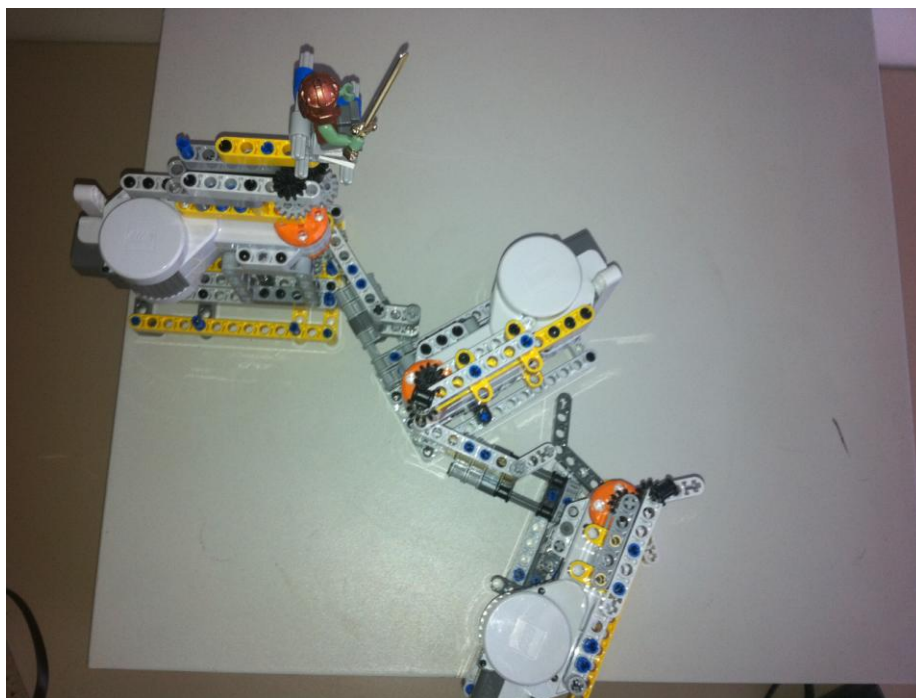
Dans McStas, ouvrir l'instrument **nxtas.instr** ou **nxtof.instr**, puis le compiler. Il suffit alors de rentrer ses paramètres et de lancer la simulation pour que McStas envoie les paramètres dans le nxt via un fichier texte temporaire, qui sera ensuite traité par le programme embarqué.

Paramètres de la simulation dans McStas.

b/ Lancer le programme

Le programme embarqué va chercher et charger le fichier texte tout au long de son exécution. Il est possible de charger de nouveaux paramètres à n'importe quel moment. Le temps de transfert est alors de l'ordre de la demi-seconde.

Comportement de la maquette : La maquette se positionne en buté (cf. photo ci-après) à chaque nouveaux paramètres détectés. Une fois le système à l'arrêt, le positionnement demandé débute. Lorsque le positionnement se termine, un signal sonore retentit.



Maquette du TAS en buté

4/ Caractéristiques de la maquette

Dimensions du support (cm) : $W \times D = 45 \times 45$

Dimensions (cm) : $W \times D \times H = 22.60 \times 33.49 \times 17.29$

Temps de réponse (s) : 0.5

Temps de positionnement moyen (s) : 5

Précision moyenne (deg) : 5-10

5/ Entretenir sa maquette

Il est nécessaire de vérifier, avant chaque utilisation, l'alignement des différents systèmes d'engrenages et en particulier celui relié directement à l'axe du moteur du monochromateur (axe qui supporte la majorité des efforts).

Bibliographie

Article sur la diffraction (LLB) : <http://www-llb.cea.fr/pedagogie/diffudiffrac/diffracentier.html>

Article wikipedia sur le NXT : http://fr.wikipedia.org/wiki/Lego_Mindstorms_NXT

NXC user manual : <http://bricxcc.sourceforge.net/nbc/nxcdoc/index.html>

McStas user manual : <http://www.mcstas.org/>

Présentation du groupe CS par Marc Johnson: <http://www.ill.eu/computing>

Articles divers.