Développement d'une interface graphique pour le logiciel de contrôle Stack-Of-Tasks

Alexandre Kuenemann

Alexandre KUENEMANN
4A (2019-2020)
Du 11 Mai 2020 au 28 Aout 2020

Rapport de stage technique du cycle ingénieur ESIEA

« Développement d'une interface graphique pour le logiciel de contrôle Stack-Of-Tasks »

Stage réalisé au sein de l'organisme

LAAS-CNRS
7 avenue de Colonel ROCHE,
31031 Toulouse
Tél : 05 61 33 62 00
France

Maitre de Stage

M Olivier Stasse
Signature :

Résumé et Abstract

Résumé

Au cours de ma quatrième année d'étude au sein de l'ESIEA, j'ai réalisé un stage en entreprise du 11 Mai au 28 Aout 2020 d'une durée de 4 mois me permettant de valider mon année d'étude.

Ce stage a été effectué au CNRS dans les laboratoires du LAAS-CNRS basé à Toulouse. J'ai été dans l'équipe GEPETTO du département robotique spécialisée dans le mouvement de systèmes anthropomorphes (robots bipèdes).

J'ai été amené à réaliser la remise en état ainsi que l'amélioration d'un plug-in python permettant de vérifier de visualiser la structure du logiciel utiliser pour générer des mouvements pour les robots humanoïdes du groupe.

Ce stage m'a permis, outre l'acquisition de connaissances utilisables dans le cadre de la robotique, de mieux appréhender le monde de la recherche et le niveau de connaissance requis dans ces secteurs.

Abstract

During my fourth year of study at ESIEA, I made a 4-month internship from May 11th to August 28th, 2020, in a company to validate my year of study.

This internship was carried out at the CNRS in the LAAS-CNRS laboratories based in Toulouse. I was in the GEPETTO team from the robotics department and specialized in the motion generation of anthropomorphic systems (bipedal robots)

I had to perform the improvement and the integration of a python plug-in. It is allowing to visualize the software graph of the control system used to generate motion for the humanoid robots of the team.

This internship allowed me, in addition to the acquisition of knowledge that can be used in robotics, to better understand the world of research and the level of knowledge required in these sectors

Table des matières

REMERCIEMENTS

Tous d'abord, je tiens à remercier mon maître de stage M. Olivier Stasse, Directeur de Recherche au LAAS-CNRS dans l'équipe GEPETTO pour m'avoir accueilli et aiguillé pendant ce stage sur mes tâches à réaliser. Grâce à sa confiance, j'ai pu accomplir avec succès les tâches qui m'ont été données pendant ces 4 mois de stage dans le laboratoire.

Je remercie aussi M. Aurélien Texier, professeur à l'ESIEA, pour son rôle de tuteur pédagogique

Je remercie très chaleureusement Messieurs Joseph Mirabel et Guilhem Saurel, appartenant eux aussi à l'équipe GEPETTO, pour leur support technique qui m'a permis de résoudre mes problèmes.

Je remercie tous les membres de l'équipe qui ont bien voulu devenir des *bêta*-testeurs de mon plug-in et me renvoyer leurs réactions.

Je tiens aussi à remercier particulièrement Arnaud Meurgues, un ami de ma famille, qui m'a permis de prendre contact avec M. Olivier Stasse au LAAS et d'obtenir ce stage.

Je tiens finalement à remercier mes parents et mon frère pour leur patience et leur support.

Introduction

J'ai effectué ce stage de 4ème année au CNRS, au sein du Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS-CNRS). J'ai été intégré à l'équipe GEPETTO centrée sur l'analyse et la génération de mouvements des systèmes anthropomorphes sous la direction de mon maître de stage : Monsieur **Olivier Stasse**.

Après une présentation de l'organisme et de l'environnement de ce stage, ce rapport présentera le contenu du stage en développant successivement les objectifs et la mission ainsi que les moyens mis en œuvre et détaillera les réalisations effectuées lors de ce stage.

Une attention particulière sera aussi donnée à la politique de développement durable et à la politique d'innovation du CNRS dans le chapitre relatif à la présentation de l'entreprise.

Enfin, je formulerai quelques conclusions quant à mes résultats, à l'acquisition de nouvelles compétences/connaissances et au ressenti de ce stage.

Présentation de l'entreprise

Le Centre National de la Recherche Scientifique (CNRS) a été fondé par décret le 19 Octobre 1939 pour mutualiser les ressources financières de la Caisse Nationale de la recherche scientifique et les activités de recherche du CNRSA.

C'est un organisme de recherche publique exerçant sous la dépendance administrative du Ministère de l'Enseignement Supérieur, de la Recherche et de l'Innovation.

Il s'agit d'un Établissement Public à caractère scientifique et technologique qui est orienté dans la recherche fondamentale et il exerce dans tous les domaines de la connaissance. Pour permettre la recherche dans tous ses domaines, le CNRS est divisé en 10 instituts avec des spécialisations uniques :

1. **L'Institut des sciences biologiques** (INSB), spécialisé en biologie.
2. **L'Institut de chimie** (INC), dont les recherches s'axent autour de la chimie du vivant et pour le vivant, la chimie verte ainsi que la fonctionnalisation de la matière.
3. **L'Institut écologie et environnement** (INEE), ayant comme domaines de recherche l'environnement et l'écologie.
4. **L'Institut des sciences humaines et sociales** (INSHS), étant plus axé sur l'homme et les cultures et sociétés
5. **L'Institut des sciences de l'ingénierie et des systèmes** (INSIS), qui privilégie l'approche « système » de ses domaines.
6. **L'Institut national des sciences mathématiques et de leurs interactions** (INSMI), ayant pour mission la coordination des recherches des différentes branches des mathématiques.
7. **L'Institut de physique** (INP), qui suit deux motivations : vouloir comprendre le monde et répondre aux enjeux de la société.
8. **L'Institut des sciences de l'information et de leurs interactions** (INS2I), qui se place au cœur des enjeux pluri et interdisciplinaires, en s'appuyant sur son partenariat avec

l'INSIS.
C'est de ces deux instituts que dépend le LAAS, l'organisme où j'ai effectué mon stage, qui sera décrit plus tard dans le rapport. L'équipe GEPETTO, dans laquelle je faisais partie, dépend quant à elle seulement de l'INS2I

9. **L'Institut national de physique nucléaire et de physique des particules** (IN2P3)
10. **L'Institut national des sciences de l'Univers** (INSU)

Ces instituts sont en pratique indépendants les uns des autres avec leurs secteurs de recherches distincts mais peuvent aussi être partenaires (partenariat INS2I / INSIS par exemple pour le groupement de recherche System On Chip, Systèmes embarqués et Objets Connectés (SOC2)).

Le rôle du CNRS est d'être utile à la société en faisant progresser la connaissance. Cette mission donnée par l'État, que le CNRS entend bien accomplir, a été divisée en 5 axes présentés ci-après :

- Faire de la recherche scientifique, grâce à ses institutions, laboratoires et sociétés partenaires,
- Valoriser les résultats pouvant satisfaire un besoin de la société ou lui en faire bénéficier
- Partager ses connaissances qui sont un patrimoine commun avec à la fois la communauté scientifique et le grand public
- Former la recherche et par la recherche et transmettre des connaissances.
- Et finalement contribuer à la politique scientifique en participant à la stratégie nationale de recherche.

Rapport d'observation sur les politiques de développement durable

J'ai effectué mon stage au LAAS-CNRS qui est une unité rattachée à l'Institut des Sciences de l'Ingénierie et des Systèmes (INSIS) du CNRS, j'aborderai donc la politique générale de développement durable du CNRS et les actions spécifiques menées par le LAAS.

Politique de développement durable au CNRS

Le CNRS étant une entreprise publique, c'est sans surprise qu'on la retrouve en tant que signataire de la « Charte développement durable des établissements publics et entreprises publiques ».

Les signataires s'engagent à définir leurs propres enjeux dans le cadre du développement durable dans le champ de leurs compétences et à les implémenter dans leur management. Ces engagements sont repris et diffués en interne et en externe via un document stratégique de développement durable et mis en place via un plan d'action.

Avec l'adoption, le 25 septembre 2015, du « Programme de développement durable 2030 » par 193 pays dont l'Etat Français, le CNRS s'est engagé à répondre à de nouveaux objectifs de développement durable.

Ce programme est organisé autour de 17 objectifs présentés dans la figure suivante.

Figure 1 : Objectifs de Développement Durable du « Programme de développement durable 2030 »

Toutes les informations relatives à ces objectifs sont accessibles sur le site du CNRS à l'adresse suivante : http://www.cnrs.fr/fr/objectifs-de-developpement-durable-le-cnrs-sengage.

Le CNRS est un organisme tentaculaire. Il n'est basé ni sur une implantation particulière (plusieurs villes et plusieurs instituts ou centres par ville), ni sur un secteur d'activité unique, ni sur un secteur de recherche et d'innovation. Tous ces instituts/laboratoires/unités propres ont une chose en commun, ils sont tous entrés dans une démarche de certification liée au développement durable, via les démarches ISO 14001 et ISO 26000.

La norme ISO 14001 a pour but la mise en place d'un management environnemental, permettant de développer un système de gestion, de fonctionnement et de production maitrisant les impacts sur l'environnement.

Cette norme fait partie de la famille ISO 14000 qui sont toutes des normes ayant un impact sur la mise en place d'un écomanagement mais ayant toutes des applications distinctes et strictes pouvant être vérifiées.

La norme ISO 26000 (Figure 2) quant à elle a pour but de mettre en place une Responsabilité Sociétale au sein d'une Entreprise (ou RSE), qui vise à réviser l'organisation de l'organisme.

D'après la commission européenne, la RSE est définie comme l'intégration volontaire par les entreprises de préoccupations sociales et environnementales à leurs activités commerciales

et leurs relations avec les parties prenantes. Une entreprise qui pratique la RSE va donc chercher à avoir un impact positif sur la société tout en étant économiquement viable.[1]

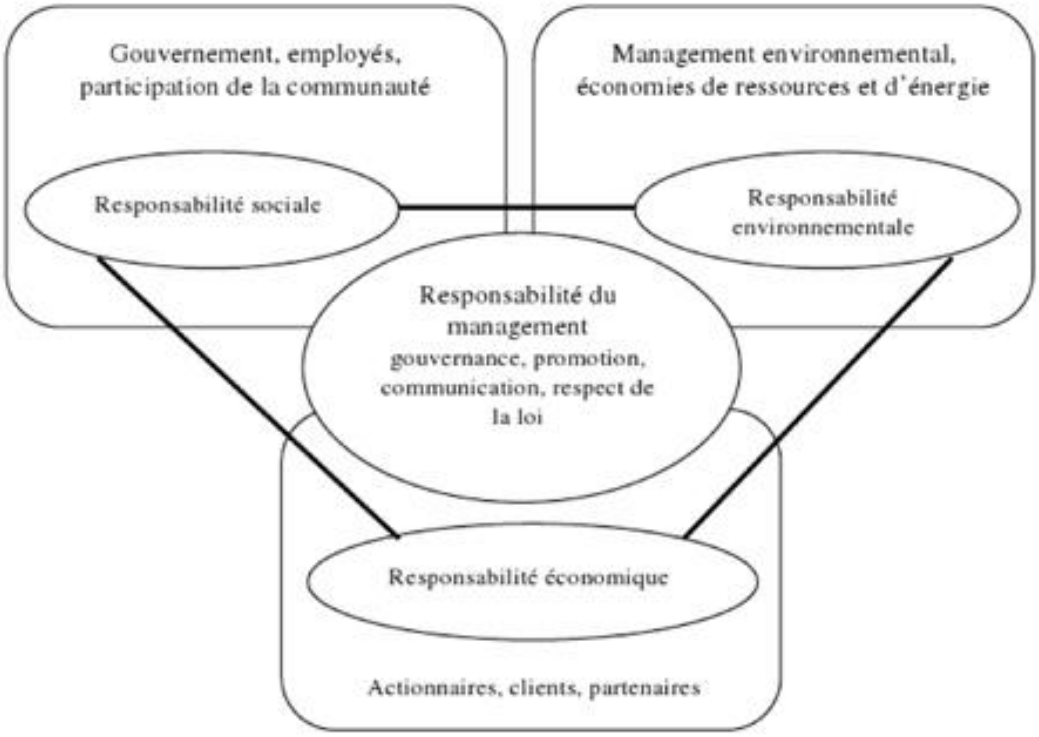


Figure 2 : Schéma explicatif de la norme ISO 26000

On peut s'apercevoir que le CNRS prend bien en compte le développement durable car une dizaine d'actions en lien avec le développement durable ont été lancées depuis 2003 et sont toujours en cours. Ces actions sont menées par des instituts appartenant au CNRS, l'Institut des Sciences Humaines et Sociales (INSHS), l'Institut Ecologie et Environnement (IEE), l'Institut National des Sciences de l'Univers (INSU), l'Institut des Sciences de l'Ingénierie et des Systèmes (INSIS) et l'Institut des Sciences Biologiques (INSB). Tous ses instituts qui n'ont pas forcément le même but peuvent se retrouver liés par ces actions, ce qui montre encore une fois que le développement durable ne se limite pas à un seul secteur d'activité.

[1] Source : https://www.economie.gouv.fr/entreprises/responsabilite-societale-entreprises-rse

Actions mises en place par le LAAS-CNRS

Dans le cadre du développement durable, le LAAS-CNRS a participé à plusieurs actions dont les principales sont :

- La création de l'article "*Multi-objective methodology to find the optimal forward current to supply Light Emitting Diode (LED) lightings*", mené conjointement entre les équipes du LAAS et du Laboratoire Plasma et Conversion d'Energie (unité mixte du CNRS, de l'Institut National Polytechnique de Toulouse (INPT) et de l'Université Toulouse 3-Paul Sabatier (UPS)). Cet article primé pendant la conférence IEEE Industry Applications Society (IAS) 2016 permet de démontrer le meilleur compromis entre consommation et coût de luminaires LED.
- La construction du "Bâtiment Expérimental" entre 2010 et 2012, maintenant appelé bâtiment Georges Giralt[2]. Ce bâtiment de recherche de 1700 m² a été construit dans une démarche de développement durable et produit plus d'énergie qu'il n'en consomme grâce aux moyens mis en place dans sa construction (plus de 700 m² de panneaux photovoltaïques pour l'électricité, un puit canadien pour le traitement de l'air couplé à des pompes à chaleur avec un haut coefficient de performance fonctionnant sur une vingtaine de sondes géothermiques).

Le LAAS a aussi mis en place un "Axe Stratégique ENERGIE" ayant pour but de mettre à profit les compétences de tous ses secteurs dans la cadre de la transition énergétique, allant de la génération et du traitement efficace de l'énergie jusqu'à la gestion intelligente de cette énergie.

[2] Georges Giralt (1930-2013) est le co-fondateur du LAAS (voir le site https://www.laas.fr/public/fr/georges-giralt-co-fondateur-du-laas-et-roboticien-visionnaire)

Rapport d'observation sur les politiques d'innovation de l'entreprise d'accueil

Avant d'aborder la politique d'innovation du CNRS et comment elle est appliquée dans le LAAS où mon stage a été effectué, je vais montrer à quelle point l'innovation est l'un des maîtres mots du CNRS.

Innovation au sein du CNRS

Le concept d'innovation ne peut voir le jour que grâce à une recherche constante d'améliorer voire de transformer un produit ou un service pour répondre à un marché.

Grâce au grand nombre de structures de recherche communes ou non avec d'autres organismes de recherche et d'entreprises (Airbus, groupe Solvay, Thales, CERN, …), le CNRS est considéré en 2019 comme la deuxième institution de recherche la plus innovante au monde derrière la Chinese Academy of Science, d'après le SCImago Institutions Rankings. Cela montre à quelle point l'innovation fait partie intégrante du CNRS.

Cette innovation aura conduit le CNRS à déposer plus de 5800 familles de brevets dont près de 30 % le sont en copropriété avec des partenaires industriels. Elle aura aussi permis depuis 1999 la création de plus de 1400 start-ups directement issues de ses laboratoires comme par exemple Greenerwave[3] et Nextmind[4] fondées respectivement en 2015 et 2017.

De plus, le CNRS soutient financièrement et conseille des projets émergents dont le potentiel d'innovation est fort, grâce à son **programme de prématuration**. Depuis sa création en 2015, le CNRS aura dépensé plus de 4 millions d'euros pour soutenir les lauréats de ce programme.

Le CNRS récompense les femmes et les hommes qui ont valorisé la recherche scientifique française en termes d'innovation depuis 2011 par la délivrance de la « Médaille de l'innovation du CNRS ». Depuis 2011, 36 personnes ont reçu cette distinction, dans des secteurs comme la robotique, la médecine, les mathématiques et d'autres encore, prouvant encore une fois que le CNRS ne se limite pas à un seul secteur de recherche et d'innovation.

[3] greenerwave.com
[4] www.next-mind.com

Politique d'innovation dans l'entreprise

Dans l'entreprise, chaque membre est intégré à une équipe de recherche et un rôle lui est donné lui définissant ses tâches. Chaque équipe travaille souvent dans un secteur de recherche différent et se spécialise dans ce domaine. Ces équipes ne travaillent pas souvent ensemble, étant donné leurs différentes spécialisations mais peuvent être liées dans un même projet quand cela le permet (ex : projet liant robotique et gestion de l'énergie).

Il y a beaucoup de collaboration au sein du CNRS, en interne entre plusieurs services d'un même ou de plusieurs instituts/laboratoires, mais aussi entre le CNRS et d'autres laboratoires et entreprises. Chaque année, des milliers de collaborations scientifiques sont encadrées par la signature de contrats de collaboration, menant à la réalisation de livrables via la création conjointe d'un programme scientifique. Pour permettre ces partenariats, plus de 140 structures de recherche sont communes entre les CNRS et les partenaires.

La prise d'initiative est aussi très appréciée, car sans initiative, il n'y aurait pas d'avancement dans la recherche, ce qui enlèverait la principale raison d'être du CNRS. Même s'il y a une ligne directrice à suivre, prendre des initiatives n'est jamais une perte de temps, cela peut aboutir sur un échec, mais même un échec n'est pas un problème, cela permet de savoir quoi éviter dans le futur. Si une idée peut être mis en place, elle sera mise en place avec des actions concrètes.

La prise d'initiative peut aussi permettre la création de nouvelles idées, nouvelles façons de voir les choses dans un projet existant ou même création d'un nouveau projet.

Enfin, le CNRS, au sein de son organigramme, dispose d'une Direction Générale déléguée à l'innovation et aux relations avec les entreprises (voir figure suivante) dont fait partie la société **CNRS Innovation**. Cette direction a pour but de diriger la politique de transferts des résultats de la recherche et les politiques de valorisations et d'interactions avec le monde socio-économique.

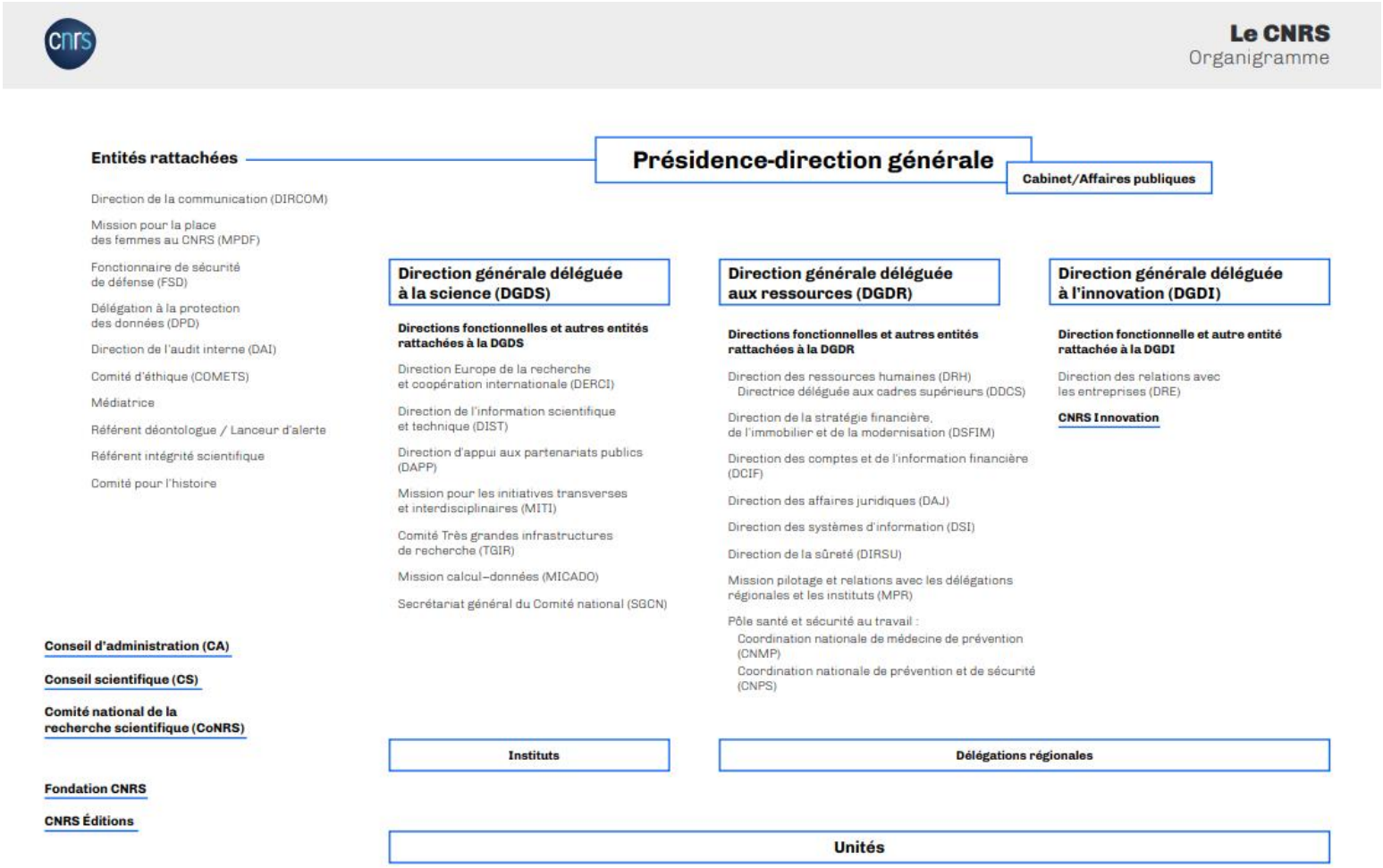


Figure 3 : Organigramme du CNRS

Environnement de stage

LAAS

Le Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS-CNRS)[5] est une unité du CNRS rattachée à l'Institut des Sciences de l'Ingénierie et des Systèmes (INSIS) et à l'Institut des Sciences de l'Information et de leurs Interactions (INS2I). Il est localisé à Toulouse.

Les 4 champs disciplinaires travaillés au LAAS sont l'informatique, la robotique, l'automatique et les micro-et nano-systèmes.

Son organisation (figure 4) est structurée de manière à permettre de développer 4 axes stratégiques :

- Intelligence ambiante
- Vivant
- Energie
- Espace.

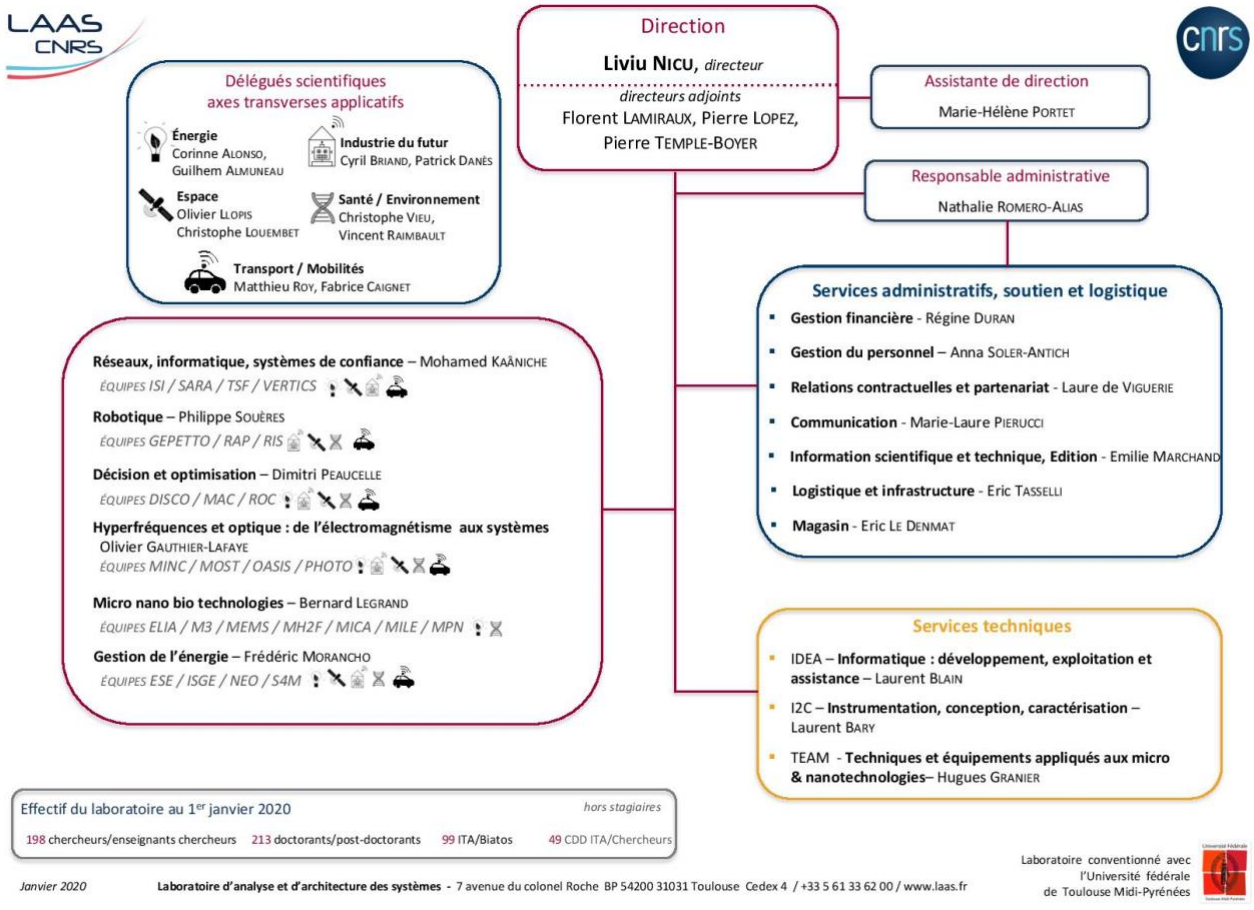


Figure 4 : Organigramme du LAAS

[5] https://www.laas.fr/public/fr/presentation-du-laboratoire

Les domaines d'application du LAAS sont très divers, allant de l'aéronautique et l'espace à la défense, en passant, entre autres, par l'agriculture et l'environnement ou les transports (automobiles ou ferroviaires) ou encore les usines du futur.

Groupe de recherche GEPETTO

Le groupe GEPETTO[6] a une activité de recherche centrée sur l'analyse et la génération de mouvement des systèmes anthropomorphes et est une équipe phare et experte en robotique humanoïde.

Le succès de GEPETTO est basé sur une approche interdisciplinaire autour de 3 objets de recherche : le robot humanoïde, le mannequin numérique et l'homme.

Dans le cadre des partenariats, le groupe GEPETTO travaille en collaboration avec AIRBUS au sein du laboratoire ROB4FAM dans le but de développer des technologies robotiques adaptées à la production aéronautique. C'est dans le cadre de ce laboratoire joint que s'est effectué mon stage.

[6]Site de présentation du groupe de recherche GEPETTO https://www.laas.fr/public/fr/gepetto et de l'équipe GEPETTO https://gepettoweb.laas.fr/

Le cadre du Stage

Objectif et Mission du Stage

L'énoncé de mon stage a été défini comme le suivant : « Développement d'une interface graphique pour le logiciel de contrôle Stack-Of-Tasks ». Ce travail s'inscrit dans l'amélioration du logiciel Stack-Of-Tasks (Mansard, Stasse, Evrard, & Kheddar, 2009) utilisé dans le groupe Gepetto pour la génération de mouvements sur le robot PYRENE (Stasse, et al., 2017). Il a été utilisé pour faire retourner une boîte au robot PYRENE de façon autonome (Nicolin, Mirabel, Boria, Stasse, & Lamiraux, 2020). Le logiciel a été utilisé aussi pour le robot HRP-2 pour du contrôle en couple (Del Prete, Mansard, Ramos, Stasse, & Nori, 2016), et pour faire tirer au robot une lance à incendie (Ramirez-Alpizar, et al., 2016). Depuis 2017, le but de mon superviseur est de tenter d'appliquer sur PYRENE (TALOS-001) les principes qui ont fonctionné sur HRP-2. Afin notamment de réaliser des interactions homme-robot (Stasse, Evrard, Perrin, Mansard, & Kheddar, 2009) et de la construction de modèle 3D d'objet (Foissotte, Stasse, Escande, Wieber, & Kheddar, 2009)

Cette interface graphique permet la visualisation du graphe de calcul du contrôleur des robots réels et simulés.

Initialement, pour visualiser ce graphe de calcul, le premier outil impliqué d'ouvrir un terminal python du robot utilisé, créer un fichier .dot contenant les informations et transformer ce fichier en fichier .pdf pour obtenir un rendu visuel. Afin d'éviter de refaire ces opérations à chaque simulation, un post-doctorant du groupe Joseph Mirabel a fait un premier prototype de plugin QT Python intégrée à l'interface graphique développée par le groupe : GEPETTO VIEWER v. 1.0.

L'objectif de ce travail est de simplifier ces opérations de visualisation du fonctionnement du robot, en temps réel, et de récupérer plus d'information par l'amélioration de ce plug-in python existant.

Dans le cadre de ce projet, mon rôle est donc d'améliorer ce plug-in Python en utilisant le feedback utilisateur pour ajouter de nouvelles fonctionnalités.

Moyens

Le travail a été effectué en utilisant les fonctionnalités du logiciel Stack-of-Tasks (SoT) et le langage Python, avec l'API QT, pour créer un plug-in QT pour l'interface graphique gepetto-viewer.

Afin de réaliser ce nouveau plug-in j'ai été amené à utiliser des logiciels complémentaires :

- GAZEBO, pour obtenir une simulation physiquement réaliste (et donc en 3D) d'un robot ;
- ROS, qui fera office de middleware entre SoT et Gazebo en permettant l'échange de données ;
- DYNAMIC-GRAPH, pour gérer le graphe de calcul qui envoie une commande au robot simulé ou réel.

Le Kit logiciel SoT, les logiciels utilisés et le langage spécifique PYQT sont décrits ci-après.

Il faut noter que je ne connaissais aucun de ces outils, à l'exception du langage Python, au début de mon stage.

Stack-Of-Tasks

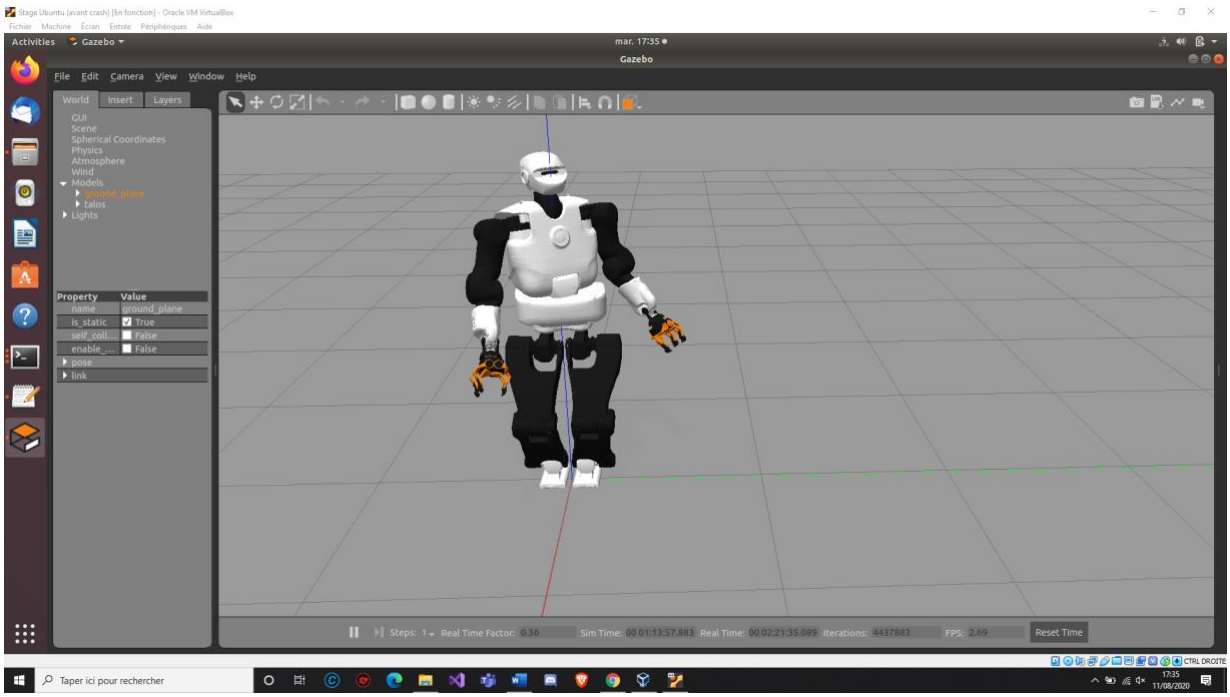
Le logiciel de contrôle Stack-Of-Tasks (SoT) est un kit de développement logiciel centré sur la robotique. Il permet de fournir des programmes pour :

- Développer des contrôleurs pour des robots complexes comme les robots bipèdes.
- Pouvoir les tester via un cadre graphique (obtenu par rviz[7], utilisable à la fois sur ROS1 et ROS2 ou gepetto-viewer), des simulateurs dynamiques (ODE[8], Bullet[9]) et un simulateur système (Gazebo).
- Implémenter SoT dans des robots réels,

Au LAAS, le kit de développement a été déployé dans deux des trois robots humanoïdes du laboratoire, HRP-2 N 14 et TALOS 001, appelé « Pyrène ».

GAZEBO

GAZEBO est utilisé pour obtenir une simulation système du fonctionnement du robot Pyrène. C'est-à-dire qu'il permet une simulation physique et une simulation de la structure logicielle sur une machine virtuelle fonctionnant sous Ubuntu installée sur mon ordinateur PC (Figure 5).



[7] Site de ROS concernant rviz : wiki.ros.org/rviz
[8] Site officiel de Open Dynamics Engine :ode.org
[9] Site de Bullet :pybullet.org

Figure 5 : Image représentant une simulation de Pyrène sous GAZEBO

Dynamic-Graph

Dans la SoT, DYNAMIC-GRAPH permet de construire un graphe de calcul en connectant des entités de calcul C++ via des signaux. Les signaux ont des dépendances temporelles entre eux. Les entités de calcul communiquent entre eux essentiellement grâce à ce mécanisme. Ceci permet de réutiliser certains sous-graphes de calcul. Le graphe de calcul peut être changé en ligne. Il est possible d'envoyer des commandes, et changer des paramètres. Depuis de nombreuses années, il avait été suggéré de faire une interface graphique permettant de visualiser ce graphe. Mon stage a consisté à réaliser ce but.

Une commande de Dynamic-Graph appelée writeGraph permet de générer un fichier sous format .dot représentant le graphe de calcul du robot tels que simulés par GAZEBO et d'en obtenir une représentation graphique montrant sa structure en temps réel sur le robot.

Ce graphique est composé d'entités qui sont tous les nœuds de calcul du robot reliés entre eux par tous leurs signaux entrants et sortants (au niveau de chaque nœud).

Dépendant des résultats voulus, on pourra observer le graphique ainsi obtenu pour vérification.

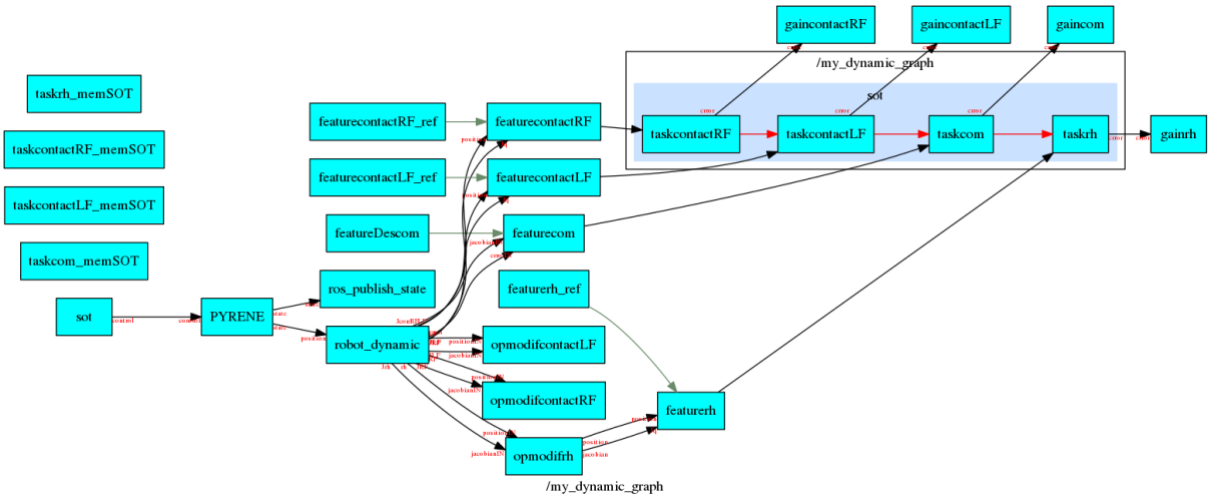


Figure 6 : Graphique obtenu par la commande writeGraph de Dynamic-Graph représentant le graphe de calcul du robot visualisé en figure 5.

ROS : Robot Operating System

ROS est un groupement de librairies et d'outils tournés sur la création d'application pour la robotique. Cet outil open source n'est utilisable que sur les environnement Linux pour la version 1. La version 2 supporte également Windows and Mac OS.

En théorie, Stack of Tasks pourrait être utilisé sans ROS mais quand on veut utiliser un robot, que ce soit en réalité ou via un simulateur, l'échange de données entre SoT et le robot serait impossible sans un intermédiaire. C'est pour cela que ROS via son API Python « rospy » va permettre cet échange d'informations.

ROS a donc seulement été utilisé en tant que middleware pour mon stage malgré son contenu bien plus conséquent.

Python

Ce langage a été utilisé dans la réalisation et l'amélioration du plug-in Gepetto viewer. J'ai utilisé l'API de QT (librairie C++) portée en python qui s'appelle PythonQT.

PythonQt, embarque un interpréteur Python dans une application C++ Qt contrairement à PyQt4/5 qui permet d'implémenter le language Python dans les librairies Qt.

La version de PythonQt utilisée est basée sur Qt4 mais ne contient pas toutes ses librairies.

Gepetto-viewer

Gepetto-viewer est une interface graphique en C++ qui se base sur open-scene-graph. Elle est accessible via un plug-in python permettant l'affichage d'un graphique similaire à celui obtenu par la SoT une fois passé en format pdf.

Il est composé de trois paquets distincts :

- Gepetto-viewer-CORBA qui fait office de serveur/client pour une interface graphique, suivant l'architecture logicielle « Common Object Request Broker Architecture » (CORBA)
- Gepetto-viewer qui sera l'interface graphique utilisé par le plug-in
- Sot-gepetto-viewer qui contient le code mis en place que j'ai remis en état de marche et amélioré

Les fonctionnalités de ce plug-in, après remise en état, étaient les suivantes :

- Création d'un graphique similaire au pdf fournit par le biais de dynamic-graph (Create entire Graph) (Figure 7)
- Zoom sur le graphique pour le meilleur affichage (Zoom fit best)
- L'arrêt de récupération de données (Stop fetching data)

- Créer un graphique centré autour d'une node contenant seulement tout le graphique pouvant être relié à ce node (Set entity filter by name) (Figure 8).

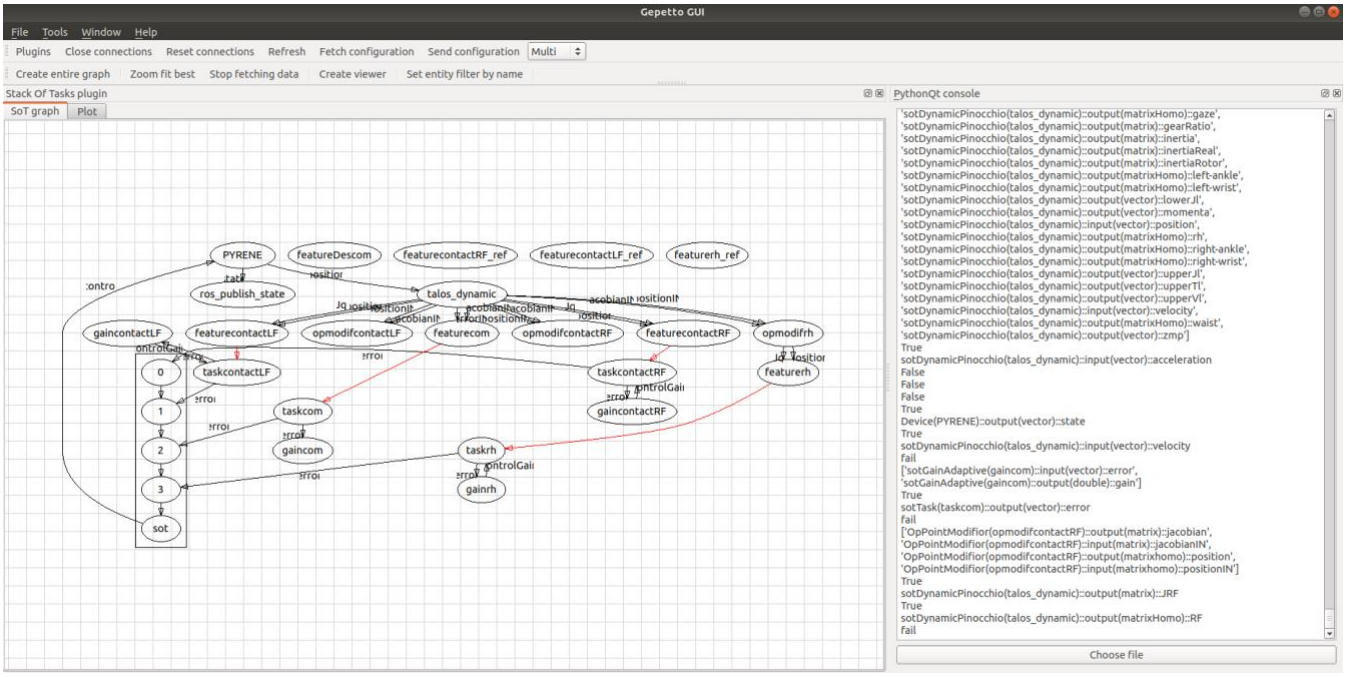


Figure 7 : Graphique équivalent à celui de la figurer 6 en utilisant Gepetto-viewer

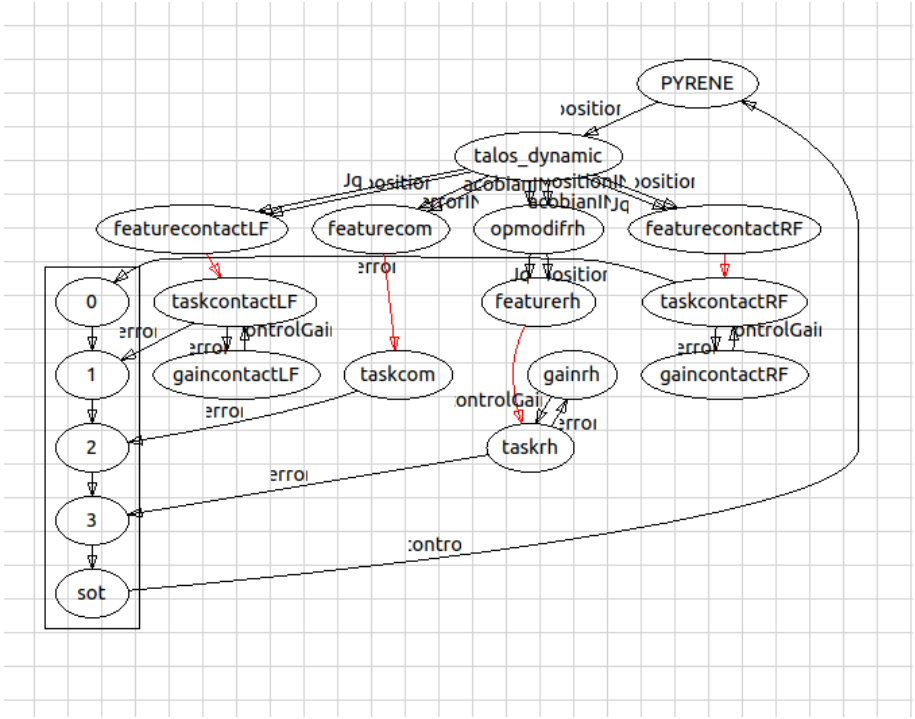


Figure 8 : Graphique centré autour du nœud « taskcom » issu de Gepetto-viewer

Réalisation

Dans cette partie, je vais décrire et expliquer les différentes actions que j'ai réalisées pendant ce stage.

Mise en place d'un environnement de travail

Problématique

À la suite du confinement lié à l'épidémie de Covid-19, seuls les chercheurs dont le travail nécessitait les outils du LAAS (manipulation sur le(s) robot(s), …) ont pu travailler sur site. Les autres, dont les stagiaires comme moi, ont réalisé leurs activités en télétravail. De ce fait, n'ayant besoin que de mon ordinateur, j'ai dû mettre en place un environnement de travail compatible avec les activités demandées.

Problèmes et Résolutions

Mon environnement de travail devait se faire sur le système d'exploitation Linux Ubuntu 18.04. En revanche, mon ordinateur personnel est un PC fonctionnant sous Windows.

J'avais donc deux solutions pour régler ce problème :

- Installer en dual boot Ubuntu, permettant de me connecter sur Ubuntu dès le lancement de l'ordinateur et d'avoir accès à toute les performances du PC.
  Pour ce faire il m'aurait fallu installer Ubuntu sur une partition de mon disque dur et stopper la connexion automatique sur Windows depuis le BIOS.
- Créer une machine virtuelle grâce à VirtualBox, me permettant de lancer une machine Linux depuis Windows.
  Contrairement au dual boot, cette machine n'a pas accès à toutes les performances du PC car une partie est bloquée par la machine Windows.

J'ai décidé de créer une machine virtuelle sur VirtualBox car j'avais déjà créé d'autres machines virtuelles et je connaissais bien son fonctionnement.

Cela m'a permis de pouvoir contrôler mon environnement Linux sans peur d'obtenir des problèmes critiques à la suite de mauvaises installations, grâce à la création d'instantanés. Je pouvais aussi stocker cette machine sur un disque dur externe me permettant d'éviter la perte de mon travail si mon ordinateur rencontrait un problème.

Même si je n'ai pas accès à toute la puissance de mon ordinateur, celle débloquée pour ma machine virtuelle restait suffisante.

A partir de cette machine j'ai pu installer les paquets de la SoT ainsi que le paquet ROS Melodic (Melodic est seulement le nom de la version) qui s'avère lui aussi nécessaire. Sans ces paquets, il m'aurait été impossible de faire des simulations du robot me permettant de tester mon plug-in.

Pour les installer, j'ai donc suivi les deux tutoriels qui ont été mis en place à ce sujet sur leurs sites respectifs[10].

Installation de gepetto-viewer

Problématique

Pour pouvoir réellement répondre à l'attente de mon stage il me fallait installer le plug-in que j'allais améliorer : sot-gepetto-viewer. Toutefois sur internet, il n'y avait aucune information pour l'installer, ni tutoriel ni paquet d'installation.

Il ne faut pas confondre gepetto-viewer avec sot-gepetto-viewer. Gepetto-viewer est le nom de l'application graphique alors que sot-gepetto-viewer est celui du plug-in.

Problèmes et Résolutions

Comme aucune information d'installation du plug-in n'existait, j'ai sollicité une personne de l'équipe au LAAS qui pouvait m'aider, Joseph Mirabel, le créateur du plug-in. Il n'existait pas de notice d'installation du plug-in, mais j'ai suivi ses instructions pour l'installation :

- Récupérer les projets « gepetto-viewer », « sot-gepetto-viewer » et « gepetto-viewer-corba » depuis leurs dépôts github respectifs.
- Installer plusieurs paquets nécessaires à la bonne installation des 3 projets précédents.
- Installer les 3 projets github grâce à la commande cmake en configurant chacune des installations avec la commande ccmake.
- Changer les variables d'environnement nécessaires

Une fois l'installation complétée, j'ai pu enfin lancer le plug-in et commencer à travailler dessus.

Pour permettre l'installation de ce plug-in par un autre utilisateur, il m'a été demandé de créer une notice d'installation de ce dernier. Cette notice décrit toutes les tâches nécessaires à l'installation du plug-in. Cette notice est visible sur mon compte GitHub[11].

Evènement : La mise à jour de Stack of Tasks

Problématique

Sachant que Stack of Tasks est un outil de développement, il n'est pas étrange qu'il y ait de nouvelles mises à jour en permanence, apportant des améliorations et parfois des erreurs. C'est ce qui est arrivé au cours du mois de Mai, un changement dans la SoT empêchait le bon fonctionnement de la simulation du robot

[10] stack-of-tasks.github.io/sot-doc/doxygen/HEAD/c_installation_detailed.html
wiki.ros.org/melodic/Installation/Ubuntu

[11] github.com/AlexKuen/sot-gepetto-viewer

Problèmes et Résolutions

Pour lancer la simulation du robot, il me fallait lancer deux scripts python.

Le premier servait à activer ROS, lancer le simulateur Gazebo, initialiser Pyrène dans le simulateur et l'afficher, lancer le contrôleur de ROS et finalement lancer Stack of Tasks.

➔ Ce script permet d'apercevoir le robot immobile dans le simulateur.

Le second, ne pouvant fonctionner que si le premier script n'avait pas de problème, vérifiait si ROS était activé. Il envoyait un signal pour faire pivoter le haut du corps du robot vers la gauche. On peut apercevoir le résultat sur la figure 6

Le problème qui est apparu avec la mise à jour empêchait le bon fonctionnement du deuxième script. Pyrène était initialisé dans le simulateur, mais il ne pouvait pas faire pivoter le robot, le programme crashait et Gazebo aussi.

J'ai remarqué, grâce à mon plug-in, que je pouvais récupérer un graphique avant que le programme ne s'arrête. J'obtenais le graphique suivant :

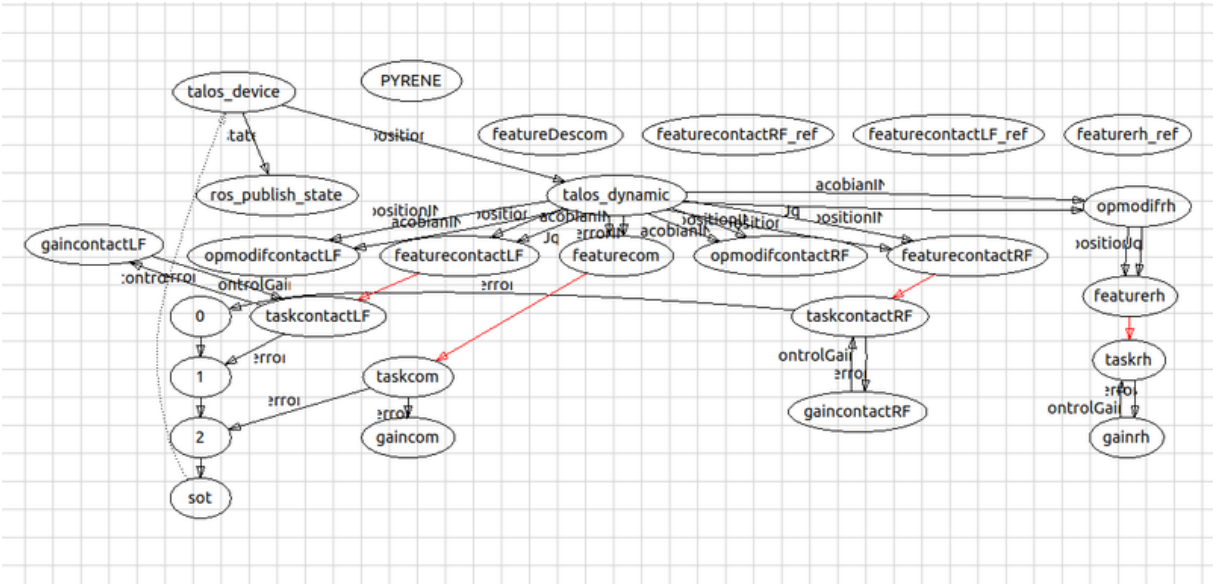


Figure 9 : Graphique généré au moment du bug

On peut remarquer dans ce graphique que le nœud « PYRENE » n'est relié à rien. Le premier script a bien créé Pyrène mais n'est pas initialisé, il initialise un robot n'existant pas qu'il a nommé « talos_device »

C'est là que vient le problème, le second script devant faire pivoter le corps de Pyrène, qui n'existe pas, ne fonctionne pas et le programme s'arrête de fonctionner.

En faisant parvenir ce graphique à mon maitre de stage, j'ai permis de faire comprendre l'origine du problème, qui a pu être résolu moins d'une semaine après sa découverte.

Mise à jour et amélioration de Gepetto-viewer

Mise à jour du plug-in

La dernière mise à jour que le plug-in avait reçu datait d'un an. Mais les paquets nécessaires à son bon fonctionnement ont continué à recevoir des mises à jour. Il n'était donc pas étonnant que le plug-in ne soit pas utilisable directement après l'avoir installé.

J'ai dû remettre en état le plug-in de la manière suivante :

- Enlever une fonction eval() qui n'avait plus d'utilité dans le fichier command_execution.py
- Remplacer des fonctions d'initialisation de variables d'une classe par des changements de variables directes via appel des variables. Le nom des variables est trouvé grâce au dictionnaire des variables de la classe en question.
  J'ai dû changer les variables de la façon suivante :
  - self.qcp.setAutoAddPlottableToLegend(True)
    → self.qcp.autoAddPlottableToLegend = True
- Remplacer : signals = self.cmd.run("[s.name for s in g.entity.Entity.entities['"+e+"'].signals()]")
  par : str_signals = self.cmd.run("[s.name for s in dg.entity.Entity.entities['"+e+"'].signals()]")
  signals = eval(str_signals)
- Avant le changement, la variable « signals » recevait un chaine de caractères où les mots étaient espacés lettre par lettre (exemple : signal lu comme s i g n a l). Ce changement permet de concaténer la chaine de caractères pour être utilisé dans une boucle for().

Amélioration du plug-in

A fin juillet, j'ai apporté les améliorations suivantes au plug-in :

1. La mise à jour automatique possible du graphique.
2. La possibilité de filtrer le graphique sur un ou plusieurs groupes de nœuds
3. La récupération d'informations concernant les signaux en lien avec un nœud.

Mise à jour automatique

La première demande d'amélioration du plug-in demandée était la possibilité de pouvoir rafraichir automatiquement le graphique s'affichant sur le plug-in, ce qui permettrait d'avoir les changements du graphique en temps réel au lieu de devoir relancer le plug-in pour afficher un nouveau graphique.

Au début, je pensais que cette amélioration allait être simple à implémenter, je pensais pouvoir utiliser une boucle while() me permettant de lancer en continu la fonction CreateAllGraph() qui est la fonction permettant la création du graphique.
Cette solution s'est avérée mauvaise. En effet, avant que le programme ne se lance, il doit être

chargé par le plug-in, mais cette boucle while() qui était chargée en continu empêchait le lancement du programme. Cette solution a donc été annulée, car bloquant le programme.

J'ai ensuite pensé à utiliser des threads : un thread pour pouvoir lancer le programme et afficher le graphique et un second pour mettre en place une boucle permettant d'envoyer un signal afin de créer un nouveau graphique.
Théoriquement, cela aurait pu marcher, mais techniquement, ce n'était pas possible. En effet, il n'y avait pas de librairie permettant la création de deux threads distincts, la seul librairie disponible en lien avec les threads s'appelle QthreadPool et ne permet que le management et la réutilisation de thread préexistant.
Cette Idée a donc aussi été abandonnée.

Finalement, j'ai trouvé le moyen de créer le rafraichissement automatique en utilisant la classe Qtimer() qui était disponible. Cette classe permet la création d'un timer qui, dans mon cas, peut permettre d'appeler une fonction en continu après un délai défini, sans pour autant bloquer le programme comme aurait fait la boucle while().
Ce rafraichissement se fait donc grâce à la fonction connect() appartenant à Qtimer. Cette fonction va permettre le lancement d'une fonction de mon choix après une durée définie avec la fonction start(). Pour cela, il me suffit de créer un timer d'une durée spécifique, signaler le lancement d'une fonction à la fin de cette durée, et quelle fonction lancer. Dans mon cas, je lance la fonction CreateAllGraph() chaque seconde, me permettant d'avoir un rafraichissement automatique du graphique chaque seconde.

J'ai donc créé la fonction LaunchRefresh() contenant la création d'un timer et le lancement automatique de la création de graphique. J'ai aussi créé une seconde fonction, StopRefresh(), pour permettre l'arrêt du rafraichissement. Quand la fonction start() permet la création d'un timer donné, la fonction stop() permet de l'arrêter, comme le timer sera stoppé, la fonction connect() ne pourra plus lancer la fonction créant un nouveau graphique. Il faudra relancer la fonction LaunchRefresh() pour avoir de nouveau une mise à jour du graphique.

Mais au lancement du plug-in, la fonction LaunchRefresh() est automatiquement lancée, il faut donc permettre à l'utilisateur de pouvoir stopper ou relancer ce rafraîchissement. J'ai dû donc rajouter dans la barre d'action du plug-in deux boutons me permettant de lancer ces fonctions, boutons appelés Launch et Stop qui vont relancer ou arrêter la mise à jour du graphique (figure 10).

Figure 10 : Image présentant les 2 boutons dans la barre d'action du plug-in

Création du Filtre

Le filtre a été la deuxième demande qui m'a été faite. Il devrait permettre de filtrer le graphique pour n'afficher qu'une partie des nœuds (tous ceux contenant « task », « rh », etc), cela peut avoir l'air d'une demande déraisonnée si l'on regarde la taille du graphique que

j'utilisais, mais ce plug-in devrait pouvoir être utilisé pour une majorité des robots du laboratoire, et avec des simulations plus complexes, résultant sur de bien plus grands graphiques. Permettre le filtrage de ces graphiques permettrait de meilleures vérifications sur des points spécifiques du robot.

Il y a eu deux versions du filtre. Un filtre simple, permettant le filtrage du graphique par un seul mot, et un multiple, remplaçant le précédent autorisant l'utilisation de multiples filtres.

Filtre Simple

Pour créer ce filtre, il fallait pouvoir récupérer un filtre donné par l'utilisateur. Pour se faire, j'ai créé un bouton dans la même barre d'action que ceux pour le rafraîchissement lançant la fonction setFilter(). Cette fonction va ouvrir une fenêtre où l'utilisateur va pouvoir renseigner le filtre qu'il veut utiliser, s'il écrit un mot, ce mot deviendra le nouveau filtre, s'il ne met rien, il n'y aura plus de filtre.

Une fois le filtre créé, il m'a fallu l'implémenter dans la création du graphique. La fonction permettant de créer le graphique fonctionnait de la manière suivante :

1. La fonction récupère une chaine de charactères contenant les noms de chacunes des entités (les nœuds),
2. Dans une boucle for() le type de chacunes des entités va être récupéré et vérifié, appartenant à Sot ou au robot. Dépendant de ce type, s'il appartient à Sot, il va diviser cette entité en plusieurs tâches et créer des nœuds pour chacune de ces tâches et les relier entre elles via leurs signaux. Si l'entité appartient au robot, le programme va seulement lui créer un nœud qui affichera son nom.
3. Une fois tous les nœuds créés, la fonction va repasser toutes les entités dans une boucle for() pour créer des liens entre des nœuds si un ou des signaux existent. Si un signal existe, une flèche sera créée sur le graphique montrant son nom ainsi que les nœuds que le signal relie.

Il faut donc appliquer le filtre sur la création des nœuds et celui des signaux.

Pour l'appliquer sur les nœuds, il m'a suffit de placer une vérification avec un if(), si le filtre est contenu entièrement ou en partie par l'entité, la création du nœud va continuer, sinon, il sera annulé car l'entité n'a pas de lien avec le filtre.

Pour créer les flèches j'ai eu besoin de deux instructions if(), une placée de la même façon que pour les nœuds, pour vérifier si l'entité en question appartient au filtre, le second après pour vérifier que le point d'arriver appartient aussi au filtre, si une flèche devait être créée entre un nœud filtré et un n'existant pas, le programme ne marcherait pas.

Ce filtre fonctionnel permettait d'afficher la partie du graphique dont le nom des nœuds contient le filtre (figure 11).

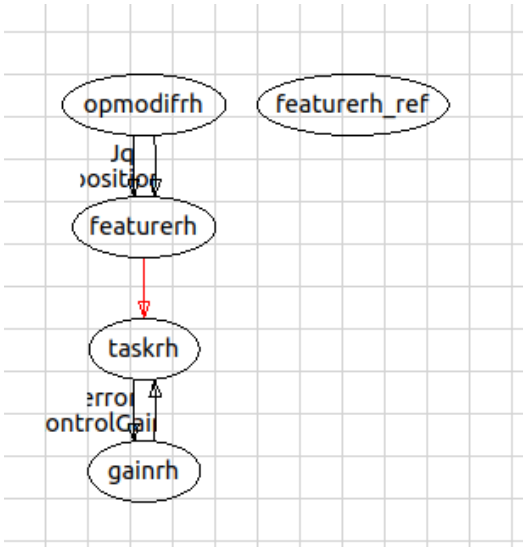


Figure 11 : Exemple de graphique utilisant le filtre « rh »

Filtre Multiple.

Ce filtre est une amélioration du filtre précédent. Au lieu de ne pouvoir renseigner qu'un seul filtre, l'utilisateur peut en spécifier plusieurs.

Avant de pouvoir faire des changements dans la fonction de création du graphique, j'ai dû changer celle permettant l'écriture des filtres. En effet, la fonction précédente ne pouvait enregistrer qu'un seul filtre.

Le bouton précédemment utilisé pour définir un filtre a été remplacé par trois boutons : « NewFilter », « Delete last Filter » et « Reset Fillter » ainsi que d'une ligne d'éditeur de texte.

La fonction « NewFilter » va permettre l'ajout du mot présent dans l'éditeur de texte dans la liste des filtres. Cette liste n'est qu'une phrase composée des filtres et d'espaces entre eux. Cette liste va ensuite être transformée en tableau de filtres avec la fonction split() qui va allouer une case d'un tableau à chacun des filtres.
La fonction « Delete last Filter » va, comme son nom l'indique, supprimer le dernier filtre qui a été rajouté à la liste en coupant cette liste grâce à la fonction rsplit(' ',1)[0]. De cette manière la chaine de charactères après le dernier espace va être supprimée.

La Fonction « Reset Filter » va quant à elle supprimer tous les filtres existants.

La ligne d'éditeur de texte va permettre à l'utilisateur d'écrire un ou même plusieurs filtres à rajouter (filtres devant être espacé par un espace) (Figure 12).

Figure 12 : Image présentant les boutons rajoutés pour le filtre multiple

Une fois les filtres récupérés, j'ai dû retransformer la fonction de création de graphique. Je ne pouvais plus vérifier la bonne utilisation des filtres avec un if() car j'aurai fait une comparaison entre une chaine de caractères et un tableau, ce qui n'est pas possible. J'ai dû donc transformer mes if() en boucles for() pour vérifier, pour chaque entité, si l'entité appartient à au moins un des filtres. Un tableau contiendra toutes les entités une fois passées pour empêcher la création d'un second nœud du même nom si plusieurs filtres laissaient passer une même entité (l'entité « taskrh » aurait créée deux nœuds si les filtres « task » et « rh » étaient enregistrés).
Ce même procédé a été utilisé pour la création des liens entre les nœuds. Ce problème aurait dédoublé un même signal (figures 13 et 14).

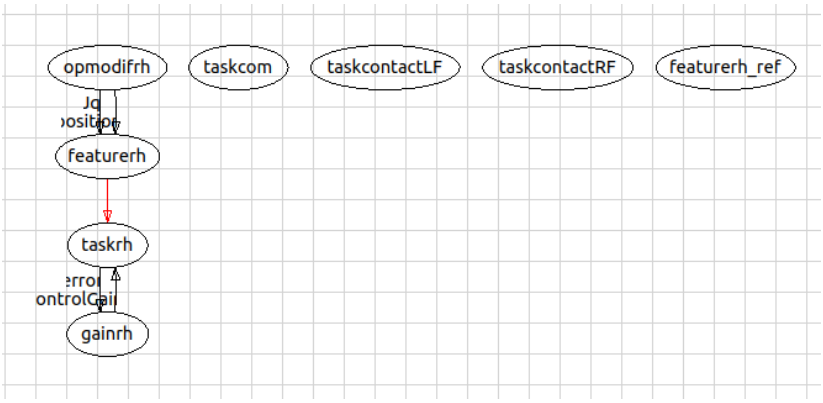


Figure 13 : Affichage du graphique utilisant les filtre « rh » et « task » (Etat normal)

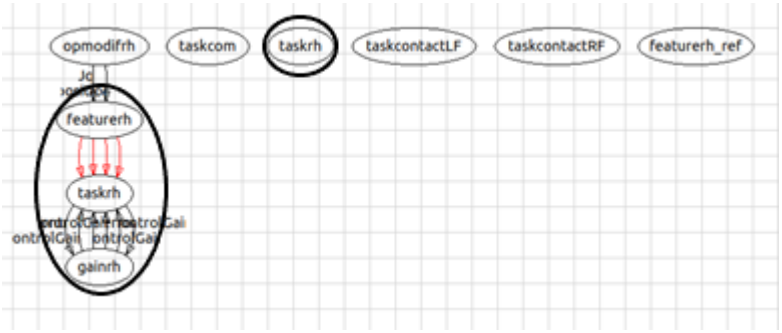


Figure 14 : Affichage du graphique utilisant les filtre « rh » et « task »
On observe un doublon du nœud taskrh ainsi que des signaux le reliant

Cette version de filtrage du graphique est fonctionnelle et répond aux attentes.

Récupération d'information

Cette amélioration m'a été demandée par soucis de visibilité du graphique, comme on peut l'observer sur la figure 7. Tous les nœuds et les signaux sont visibles et presque tous lisibles. Mais plus le graphique devient grand et plus il y a de signaux, moins ces derniers sont lisibles (Figure 15).

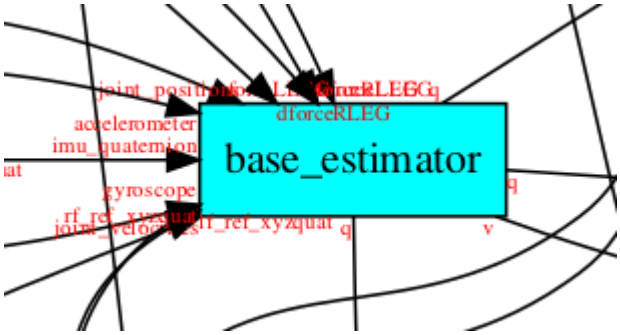


Figure 15 : Exemple d'un fragment de graphique contenant beaucoup de signaux

On m'a donc demandé, pour répondre à ce problème, de pouvoir afficher tous les signaux entrants et sortants concernant un nœud selon la volonté de l'utilisateur.

Étant donné que la fonction de création du graphique a besoin des signaux pour pouvoir les nommer sur le graphique, j'ai pu réutiliser la même fonction pour récupérer les informations dont j'avais besoin. Il s'agit de la fonction « self.cmd.run("[s.name for s in dg.entity.Entity.entities['"+e+"'].signals()]") »
qui permet de récupérer tous les signaux ayant un lien avec une entité « e ». Au début, je pensais que lister tous ces signaux avec un print() aurait été suffisant, mais ce n'était pas pratique et si le rafraîchissement était activé, les informations voulues n'étaient plus visibles dans le terminal.

J'ai donc réglé ce problème en utilisant une fenêtre d'information en utilisant une QMessageBox.information. Cette fenêtre contient tous les signaux liés à une nœud, ainsi que leur type. Elle peut être ouverte après un clic droit sur un nœud (figure 16).

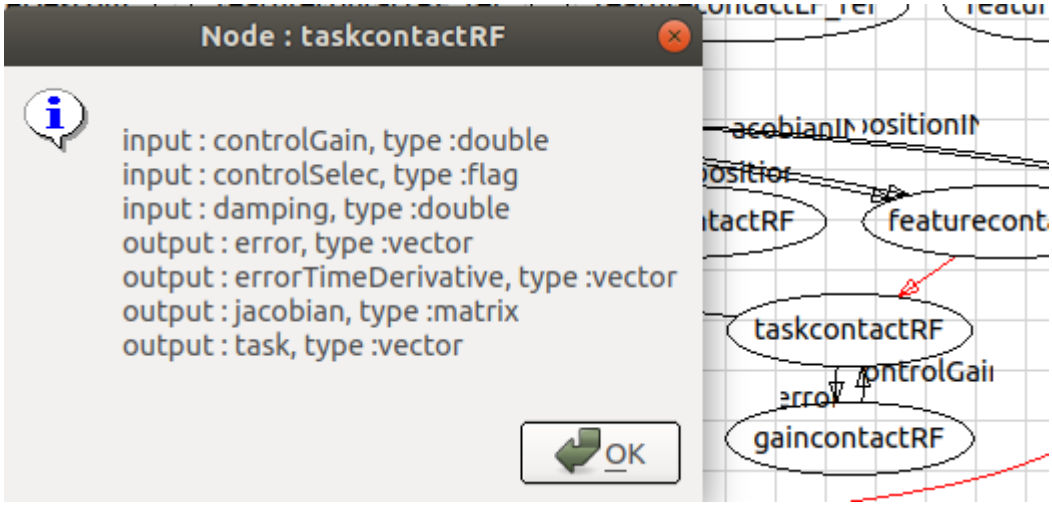


Figure 16 : Exemple de récupération des informations des signaux appartenant au nœud « taskcontactRF »

Conclusions

Résultats et voies d'amélioration

À la suite de ce travail et à la réalisation de ce rapport, j'ai implémenté dans le plug-in un rafraichissement automatique, la possible utilisation d'un filtre ainsi que la récupération d'informations de signaux par rapport à un nœud.

Mais d'autres améliorations pourront encore être implémentées d'ici la fin de mon stage. Une en particulier permettra l'affichage des différentes dépendances utilisées par le gepetto-viewer.

Ce plug-in pourra continuer d'être amélioré par de nouvelles personnes, chercheurs ou stagiaires comme moi.

Acquisitions personnelles

Sur le plan technique

Ce stage m'a permis de développer de nouvelles compétences, grâce à l'utilisation de nouveaux logiciels et de langage informatique que je n'avais jamais utilisés. Il m'a aussi permis d'améliorer ma compréhension de l'utilisation du langage python ainsi que sur les systèmes d'exploitation Linux et en particulier la version Ubuntu 18.04.

Ce stage m'a aussi permis d'aborder le monde de la recherche en robotique en collaborant, à mon échelle, au développement de robots humanoïdes au sein de l'équipe GEPETTO. J'ai pu

appréhender le fonctionnement d'un tel robot et des problématiques qui sont liées à son fonctionnement.

J'ai pu affiner mon sens de l'organisation, l'équipe mettait en place des réunions journalières pour voir l'état d'avancement de chacun des membres du groupe, la réunion du Lundi permettant de faire un point sur ce qu'il s'était passé la semaine précédente. Cela m'a permis de définir mes actions semaine par semaine et de suivre cette ligne directrice et les échéances fixées.

Sur le plan personnel

Malheureusement, à la suite de l'épidémie du COVID-19, j'ai été obligé de travailler depuis chez moi pendant toute la durée de mon stage. Ceci m'a empêché de pouvoir travailler au LAAS à Toulouse et de rencontrer physiquement les membres de l'équipe.

Cela ne m'a pas permis de bénéficier de ces contacts rapprochés avec les autres stagiaires et les thésards avec une ambiance qui alliait bonne humeur et productivité.

Je n'ai pas eu la possibilité de participer aux activités sociales et extra professionnelles qui ont été mises en place par les membres de l'équipe.

Travailler depuis chez moi a aussi été plus compliqué que je ne l'aurais pensé, Contrairement à un travail dans les bureaux/locaux d'une entreprise, le travail à domicile procure plus de distractions et j'ai dû organiser mon temps de travail sur la journée afin de maintenir ma concentration, mon attention et donc ma productivité.

J'ai pu aussi devenir plus autonome. En effet, j'étais à la fois le seul en charge de l'amélioration du plug-in et en même temps je travaillais seul en télétravail chez moi, m'obligeant à trouver les solutions aux problématiques, à les mettre en œuvre et à prendre moi-même des décisions.

Je pense que ce stage a aussi été un bon choix pour moi et confirme mon choix de la mineure Robotique pour ma 5ème année d'étude.

Bibliographie

Del Prete, A., Mansard, N., Ramos, O., Stasse, O., & Nori, F. (2016). Implementing torque control with high-ratio gear boxes and without joint-torque sensors. *International Journal of Humanoid Robotics*.

Foissotte, T., Stasse, O., Escande, A., Wieber, P., & Kheddar, A. (2009). A two-steps next-best-view algorithm for autonomous 3d object modeling by a humanoid robot. *IEEE International Conference on Robotics and Automation.*

Mansard, N., Stasse, O., Evrard, P., & Kheddar, A. (2009). A Versatile Generalized Inverted Kinematics Implementation for Collaborative Working Humanoid Robots: The Stack of Tasks. *Int. Conf. on Autonomous Robots.* IEEE.

Nicolin, N., Mirabel, J., Boria, S., Stasse, O., & Lamiraux, F. (2020). Agimus: a new framework for mapping manipulation motion plans to sequences of hierarchical task-based controllers. *IEEE/SICE International Symposium on System Integration (SII).*

Ramirez-Alpizar, I., Naveau, M., Benazeth, C., Stasse, O., Laumond, J.-P., Harada, K., & Yoshida, E. (2016). Motion generation for pulling a fire hose by a humanoid robot. *IEEE-RAS 16th International Conference on Humanoid Robots*.

Stasse, O., Evrard, P., Perrin, N., Mansard, N., & Kheddar, A. (2009). Fast foot prints re-planning and motion generation during walking in physical human-humanoid interaction. *IEEE-RAS International Conference on Humanoid Robots.*

Stasse, O., Flayols, T., Budhiraja, R., Giraud-Esclasse, K., Carpentier, J., Del Prete, A., . . . Ferro, F. (2017). TALOS: A new humanoid research platform targeted for industrial applications. *Int. Conf. on Humanoid Robotics.* IEEE.

Evan Drumwright, John Hsu, Nathan P. Koenig, Dylan A. Shell (2010): Simulation, Modeling, and Programming for Autonomous Robots - *2nd Int Conf, SIMPAR.*