Julien Vinel
Baptiste Bolzer
Vincent Porziemsky
Clément Folny

Projet long N7 2019

Simulation et commande d'un réseau de transport et un système de production robotisés

Encadré par Sandra Ngueveu et Cyril Briand

Remerciements

Nous tenons à remercier Madame U. Ngueveu Sandra ainsi que Monsieur Briand Cyril pour nous avoir permis de réaliser ce projet et pour l'aide et l'encadrement qu'ils nous ont donnés.

Nous tenons à remercier également les équipes de l'AIP-PRIMECA Toulouse pour l'accueil qu'ils nous ont offert.

Table des matières

Introduction

Dans le cadre de notre formation d'ingénieur à l'ENSEEIHT en option CDISC (Commande, Diagnostique et Informatique des Systèmes Critiques), en 3ème année nous avons l'opportunité de réaliser un projet long de 6 semaines. Nous avons réalisé le nôtre au sein de l'AIP-PRIMECA.

Ce projet a pour but de réaliser la simulation et la commande d'un réseau de transport d'un système de production disponible à l'AIP, en vue du prochain remplacement du TER atelier flexible. Dans notre cas, il s'agit d'un réseau monorail sur lequel se déplacent des navettes. Celui-ci est également constitué de quatre stations robotisées permettant le traitement des pièces transportées.

De précédents projets longs ont permis d'obtenir une simulation 3D de la cellule Montrac et une interface homme-machine affichant l'état des capteurs et actionneurs de la maquette en temps réel.

Notre premier objectif est d'accélérer la vitesse de la simulation, qui était trop lente et trop contraignante à utiliser. La simulation était également trop éloignée du fonctionnement de la maquette, et d'un sujet de TER. Nous allons faire évoluer la couche haute afin de faciliter le futur passage de la simulation à la maquette. Durant ce projet long, nous allons notamment travailler avec l'outil ROS et le logiciel V-rep.

Présentation de l'AIP-PRIMECA

Réparti sur tout le territoire et composé de 9 centres régionaux, le réseau AIP-PRIMECA est un réseau académique de chercheurs et d'enseignants chercheurs associé à des moyens technologiques de haut niveau pour la mise en commun de moyens techniques.
Il est le résultat de la fusion entre :

- les A.I.P. (Ateliers Inter-établissements de Productique), créés en 1984 et utilisées comme support expérimental de formations approfondies dans le domaine de la Productique.
- PRIMECA (Pôles de Ressources Informatiques pour la MECAnique), créés en 1991 dans le but de promouvoir l'utilisation des outils informatiques dans la conception des produits mécaniques.

Les établissements toulousains (Université Paul Sabatier, INP Toulouse, INSA Toulouse, LAAS-CNRS) ont développé dès 1983 une politique de site afin de renforcer la formation pratique dans certains domaines nécessitant des moyens lourds et coûteux, en phase avec la réalité industrielle. Cette politique de mutualisation des ressources et des compétences s'est traduit pas la création de trois ateliers interuniversitaires dont l'AIP-PRIMECA, dans les domaines de la Mécanique et de la Productique.

Présentation de la cellule flexible

L'AIP-PRIMECA de Toulouse met à disposition de nombreux moyens dont la cellule flexible de production robotisée MONTRAC. Elle est constituée d'un réseau de transport monorail, appelé MONTRAC, sur lequel circulent des navettes. Ces dernières sont alimentées par les rails et sont donc en mouvement dès que la cellule est alimentée. Elles sont aussi autonomes et intelligentes : en effet, le seul moyen de les contrôler est de commander les actionneurs placés sur les rails via des automates.

Afin d'empêcher toute collision, chaque navette possède un capteur de proximité frontal qui permet de la stopper lorsqu'elle est trop proche d'une autre. La figure ci-dessous présente un schéma de la cellule :

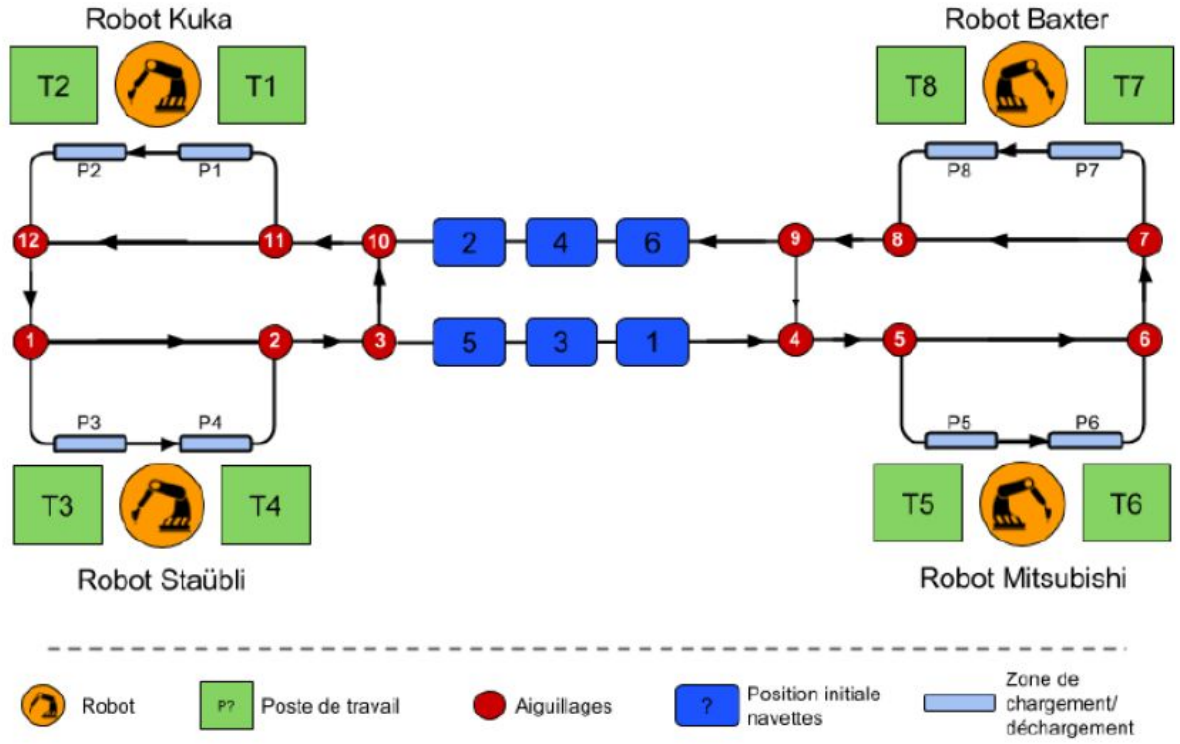

Figure 1 - Structure de la cellule flexible robotisée

La cellule est composée de 4 zones de travail (chacune séparée en 2 zones de chargement/déchargement représentée en bleu clair et 2 postes de travail en vert) sur lesquelles 4 robots réalisent des opérations sur les produits transportés par les navettes. Ces zones sont desservies par des monorails en aluminium sur lesquels les navettes circulent, tout en respectant le sens de circulation indiqué par les flèches noires. Ce sens unique de circulation est imposé par l'alimentation latérale des rails et permet d'éviter les risques de collisions frontales. Le routage est réalisé grâce aux 12 aiguillages présents (A1 à A12 représentés par des cercles rouges sur la figure 4). Le

tout est divisé en 5 zones (ici invisibles), contrôlées chacune par un automate programmable

Capteurs et actionneurs

Les capteurs permettent de connaître la position des navettes, des aiguillages et des ergots. Les ergots servent à bloquer les navettes au niveau des zones de chargement/déchargement, ce qui permet aux robots de manipuler les produits positionnés sur les navettes, sans risque de les déplacer accidentellement. Les actionneurs permettent de commander les points d'arrêts des navettes, les aiguillages et la position des ergots. La liste des capteurs et actionneurs est donnée ci-dessous, ainsi qu'un schéma de leur répartition dans la cellule :

RxG	Positionner l'aiguillage x à gauche
RxD	Positionner l'aiguillage x à droite
Dx	Déverrouiller l'aiguillage x
Vx	Verrouiller l'aiguillage x
STx	STx = 0 arrête la navette au niveau de l'actionneur (= 1 libère la navette)
PIx	Blocage/Déblocage des navettes sur la zone de chargement

Tableau 1 - Actionneur du système

CPx	Capteur de position, vaut 1 quand une navette est sur le capteur
PSx	Capteur de stop situé en face d'un actionneur STx pouvant arrêter la navette
CPIx	Vaut 1 quand l'ergot PI est sorti
DxD	Vaut 1 quand l'aiguillage est à droite
DxG	Vaut 1 quand l'aiguillage est à gauche

Tableau 2 - Capteurs du système

Voici un schéma de la cellule flexible (présent sur l'interface utilisateur) mettant en évidence les différents capteurs et actionneurs cités précédemment :

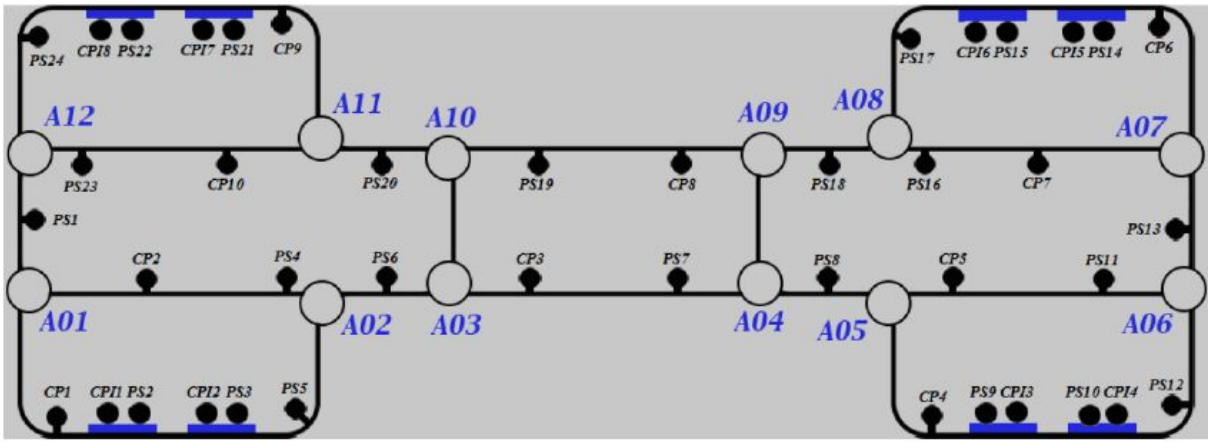

Figure 2 - Structure des aiguillages et différents capteurs

Les actionneurs STOP/START (STx), non visibles ici, permettent de bloquer les navettes et sont placés avant les aiguillages et au niveau des postes. Les actionneurs PIx font sortir l'ergot, permettant de stabiliser la navette au niveau des postes de travail.

Il est pertinent de préciser que les ergots n'arrêtent pas physiquement la navette mais font office d'objet de détection magnétique pour la navette.

De plus, à chaque aiguillage sont associés deux capteurs DxD et DxG qui permettent de détecter sa position (respectivement droite ou gauche).

Objectif final: l'usine 4.0

À l'heure actuelle, l'utilité de cette maquette est de pouvoir développer des Travaux Pratiques destinés à des étudiants de cycle supérieur issus de différentes formations de Toulouse, en particulier pour l'ENSEEIHT qui souhaiterait remplacer une ancienne maquette (TER atelier flexible) qui ne fonctionne plus. Ces TP devraient permettre de former les étudiants à la commande de haut niveau de systèmes de production en respectant un ordonnancement préétabli et les gammes de production. Elle permettra aussi de se confronter et de se familiariser avec les notions de ressources partagées au sein d'un réseau de pétri complexe. Ce projet s'inscrit dans la problématique de l'usine du futur, dans le sens où l'ensemble des différents capteurs présents sur la cellule permettent une communication constante entre les machines et les humains. En effet, il y a maintenant plusieurs années que l'on commence à parler d'usine du futur, ou « usine 4.0 », capable de faire communiquer les machines entre elles. Son principe est qu'à chaque maillon des chaînes de production et d'approvisionnement, les outils et postes de travail communiquent en permanence grâce à Internet et aux réseaux virtuels. Machines, systèmes et produits échangent de l'information, entre eux ainsi qu'avec l'extérieur". Dans notre cas, il y a une communication constante entre l'utilisateur et la cellule, mais également une communication constante entre chaque navette, les rails et les robots. Il devient alors beaucoup plus aisé de localiser la source d'une panne et de rediriger la production sur d'autres machines afin d'éviter l'arrêt total de la production durant le temps de manutention de la machine en panne. Nous produisons ainsi un suivi en temps réel de l'avancement des différentes tâches de la production et de l'acheminement, ce qui permet d'avoir une gestion et une planification de la production plus performante.

Organisation du travail : méthode agile

Pour pouvoir s'organiser entre nous quatre et pouvoir laisser une trace au jour le jour des progrès réalisés à nos tuteurs nous avons choisis de mettre en place une méthode d'organisation agile. Cette méthode demande une communication constante entre les clients (ici nos tuteurs) et les développeurs, ainsi nous nous somme rencontrés

tous les vendredis matin et nous avons mis en place un outil de gestion de projet interactif en ligne avec le site Trello. Un Trello permet de fixer des tâches sous forme de post it dont la coloration indique l'avancement.

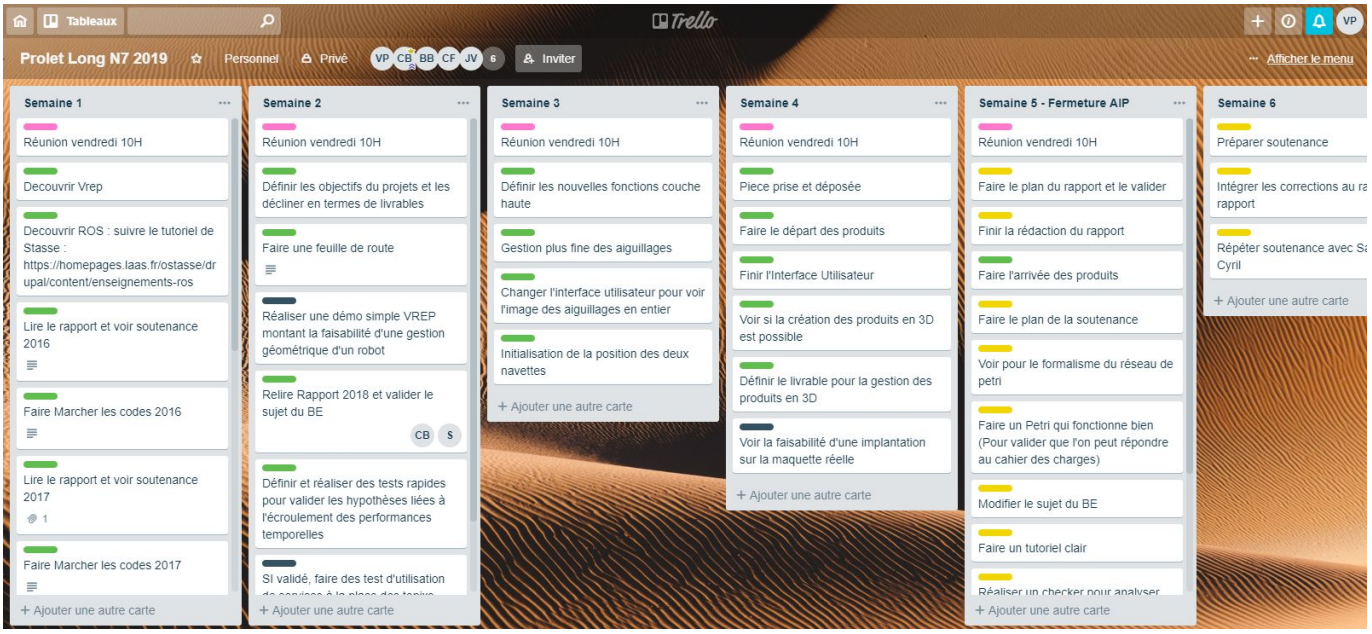

Figure 2 -Le Trello mis en place lors du projet

Etat des lieux et objectifs

Lorsque nous sommes arrivés, nous avons récupéré les anciens fichiers des projets longs 2016, 2017, et 2018. La première chose à faire était donc de s'approprier les différents fichiers, de bien les prendre en main, il a également fallu se former sur l'outil ROS et le logiciel V-REP qui vont nous servir tout au long du projet afin de travailler sur cette simulation. Nous avons réalisé un état des lieux de l'avancement du projet afin de définir les objectifs que nous devons atteindre cette année. Tout d'abord, nous avons eu des difficultés à lancer simplement la simulation, il serait intéressant d'écrire un tutoriel principal très détaillé. Après avoir lancé la simulation, la première chose que l'on remarque est la lenteur de cette dernière, à telle point que pour l'instant, celle-ci n'est pas exploitable pour un TER. De plus, l'interface est perfectible : certains boutons sont inutiles ou ne correspondent pas à ce qu'on aura besoin pour la simulation finale. Les fonctions couche-haute sont également à retravailler : les navettes sont dotées d'une "intelligence", les aiguillages sont ainsi gérés automatiquement par ces dernières, mais nous souhaitons pour le futur sujet de TER que les aiguillages soient gérés par les étudiants via leur réseau de pétri. De plus, il était trop compliqué de créer des objets 3D pour représenter les produits qui seront transportés par les navettes. Les étudiants précédents ont donc fait le choix de changer la couleur des navettes et des postes de travail suivant les produits transportés.

De ces observations nous pouvons définir nos principaux objectifs : tout d'abord nous devons identifier la raison pour laquelle la simulation est si lente, et résoudre ce problème. Ensuite nous aurons à retravailler le fonctionnement de la simulation, et modifier les fonctions en couche haute afin que ces fonctions correspondent mieux aux besoins pédagogiques du TER. Nous allons également revoir l'interface. Etant donné le choix qui a été pris concernant la coloration des tables et des navettes plutôt que de gérer des produits en 3D, il est intéressant de coder un "checker" qui vérifiera le bon fonctionnement de la solution proposée par l'étudiant, et précisera les éventuelles erreurs. Enfin, nous pouvons également commencer à travailler sur le futur de la simulation, à savoir la gestion d'objets en 3D. Finalement nous réaliserons sujet de TER à partir du programme existant afin de fournir un livrable complet et prêt à l'utilisation par les élèves de l'ENSEEIHT. Nous devions donc penser à un sujet complet permettant d'illustrer toutes les problématiques voulues liées au programme pédagogique en place dans la filière GEA.

Montée en compétence sur les logiciels

Robot Operating System

Pour avancer dans notre projet, deux outils seront primordiaux.

Le premier est le middleware ROS (Robot Operating System), une plateforme de développement logiciel qui fournit des bibliothèques et des outils pour aider les développeurs de logiciels à créer des applications robotiques. Pouvant fonctionner sur un ou plusieurs ordinateurs, il procure de nombreuses fonctionnalités telles que l'abstraction du matériel, le contrôle des périphériques de bas niveau, la transmission de messages entre les processus et la gestion des packages installés. De manière simple, ROS permet de créer des sous-programmes appelés nœuds ou nodes en anglais, qui peuvent communiquer entre eux à l'aide de messages synchrones ou asynchrones. ROS offre une architecture souple de communication interprocessus et inter-machine. Dans le cas de messages asynchrones, les processus ROS nodes, peuvent communiquer avec d'autres via des topics.

La connexion entre les nodes est gérée par un master et suit le processus suivant :

- Un premier node avertit le master qu'il a une donnée à partager
- Un deuxième node avertit le master qu'il souhaite avoir accès à une donnée
- Une connexion entre les deux nodes est créée
- Le premier node peut envoyer des données au second

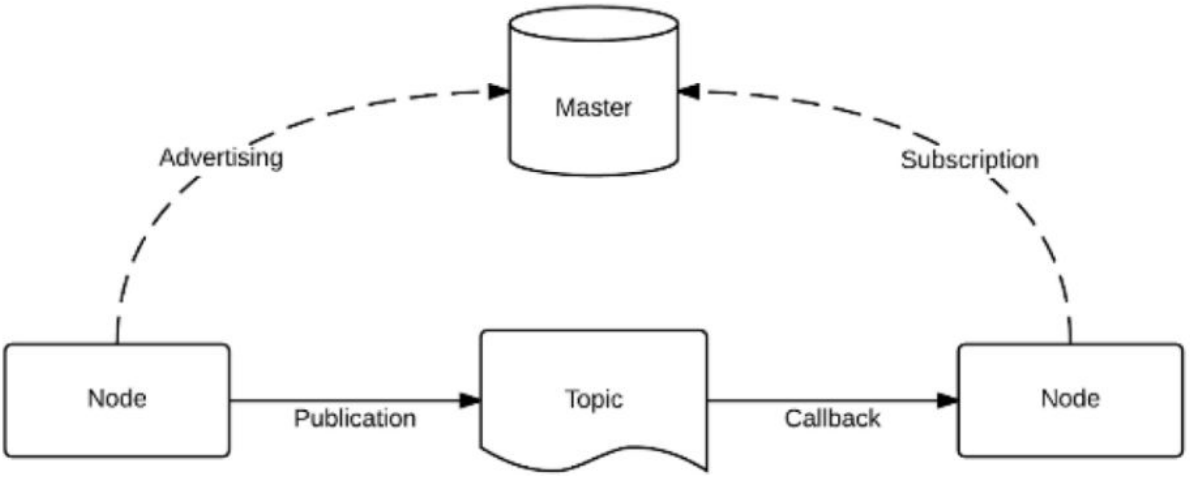

Figure 3 : Fonctionnement de ROS

Un node qui publie des données est appelé un publisher, et, un node qui souscrit à des données est appelé un subscriber. Un node peut être à la fois publisher et subscriber. Les messages envoyés sur les topics sont, pour la plupart, standardisés, ce qui rend le système extrêmement flexible.

Pour les messages synchrones, que l'on appelle services, les nœuds ont seulement accès aux services offerts par chacun d'entre eux, par l'intermédiaire de clients. Sans entrer dans le détail, les entrées et sorties de chaque service sont définies au niveau des nœuds où ils sont instanciés, et elles sont connues des nœuds clients correspondants. Il est important de remarquer que ROS permet une communication inter-machine, des nodes s'exécutant sur des machines distinctes, mais ayant connaissance du même master peuvent communiquer de manière transparente pour l'utilisateur. Ceci représente une des grandes forces de ROS. De plus, ROS est sous licence open source, ce qui permet d'avoir accès à une grande communauté et de disposer de nombreux tutoriels. ROS est aujourd'hui, officiellement supporté par plus de 75 robots. La grande souplesse de ROS lui permet d'être déployé sur des robots très différents (robot mobile, bras industriel, multicoptère) et qui évoluent dans des milieux variés (terrestre, aérien, marin et sous-marin).

Logiciel de simulation VREP

Le second outil que nous avons utilisé est V-REP, un simulateur dédié à la simulation de scènes composées de robots et développé par la société Coppelia Robotics. L'intérêt de ce simulateur est qu'il est compatible avec ROS. En d'autres termes, il est vu par nos nœuds ROS comme un nœud spécifique, avec lequel nous pouvons communiquer.

Il permet la simulation de la cellule MONTRAC, se situant à l'AIP. La scène a été préparée par les groupes précédents. La simulation V-REP finale permet donc de faire

évoluer les navettes sur les rails, tout en communiquant l'état des capteurs au reste de notre simulateur, grâce à son nœud ROS.

Accélération de la simulation

Le premier objectif est primordial, cela concerne l'accélération de la simulation qui reste trop lente pour pouvoir être utilisée en TER par les étudiants. Cela empêcherait les étudiants de travailler dans de bonnes conditions, mais aussi perturberait leur progression vu le temps restreint dont ils disposent. Il fallait donc trouver un moyen d'augmenter la vitesse de simulation pour rendre les tests plus fluides.

Après être montés en compétences et avoir étudié les codes des années précédentes, nous avons émis quelques hypothèses concernant la raison d'une simulation si lente.

Tout d'abord, nous pensions que cela était dû au très grand nombre de topics utilisés, en effet pour chaque poste, chaque tâche, chaque robot il y a un topic mis en place pour transmettre et recevoir différentes informations, or cela utilise beaucoup de ressources. Ainsi nous avions pensé à remplacer les topics par des services car ceux-ci sont asynchrones contrairement aux topics qui sont synchrones.

La deuxième idée correspondait à regrouper certains topics en un seul, en effet, au lieu d'avoir plusieurs topics par poste, nous n'en aurions qu'un seul.

Après plusieurs tests, notre première hypothèse était devenu caduque et la seconde impliquait une restructuration générale de l'architecture des topics utilisés. En étudiant la structure des topics et la manière dont les groupes précédents avaient paramétré les topics, nous avons alors découvert dans les fichiers commande_locale.cpp, main_robotX.cpp (X le numéro du robot) et main_tacheX.cpp (X le numéro de la tâche) que des topics étaient mal paramétrés. En effet voilà comment fonctionne normalement une lecture de topic: les messages sont stockés dans une liste qui est lue par la fonction ros::spinOnce une fois par période, le reste du temps cette fonction dort. Ici, la commande loop_rate.sleep() n'était pas appelée et le programme ne dormait pas, on regardait la liste en continu ce qui consommait énormément de ressources et ainsi ralentissait la simulation. Il a donc fallu corriger cet oubli.

Dans les fichiers .cpp cela se code sous cette forme:

```
ros::Rate loop_rate(f);  /// f la fréquence choisie
while(ros::ok())
{
        ros::spinOnce();
        loop_rate.sleep();
}
```

Et voici le choix des différentes fréquences :

- commande_locale.cpp ----------> f=30Hz
- main_robotX.cpp ----------> f=25Hz
- main_tacheX.cpp ----------> f=25Hz

Après modification, la simulation était en effet beaucoup plus rapide et donc utilisable par les étudiants pendant le TER.

Interface Utilisateur

L'interface utilisateur est la partie vivante de la simulation, elle permet d'afficher la simulation 3D V-rep en temps réel et d'influer sur cette modélisation à l'aide de boutons. Au départ cette interface avait quelques défauts qu'il a fallu résoudre:

- Premièrement la taille de l'interface était trop grande et elle ne s'affichait pas en entier sur l'écran (ce qui est important car il faut pouvoir voir l'affichage des aiguillages). Il a fallu modifier la taille à l'ouverture ainsi que permettre la modification de cette taille avec la souris pour que cela puisse s'adapter à tout ordinateur, ainsi l'utilisateur a une meilleure visibilité totale.
- Puis il a fallu enlever des boutons qui étaient obsolètes, notamment un bouton "TerOn" qui n'avait aucune utilité et un bouton "Add Shuttle" qui n'a plus lieu d'être car le nombre de navettes doit être fixe du lancement à la fin de la simulation (conformément à la maquette réelle).
- Nous avons aussi ajouté un bouton "Add Product" qui permet, dans le mode manuel, d'ajouter des produits sur le poste 2.

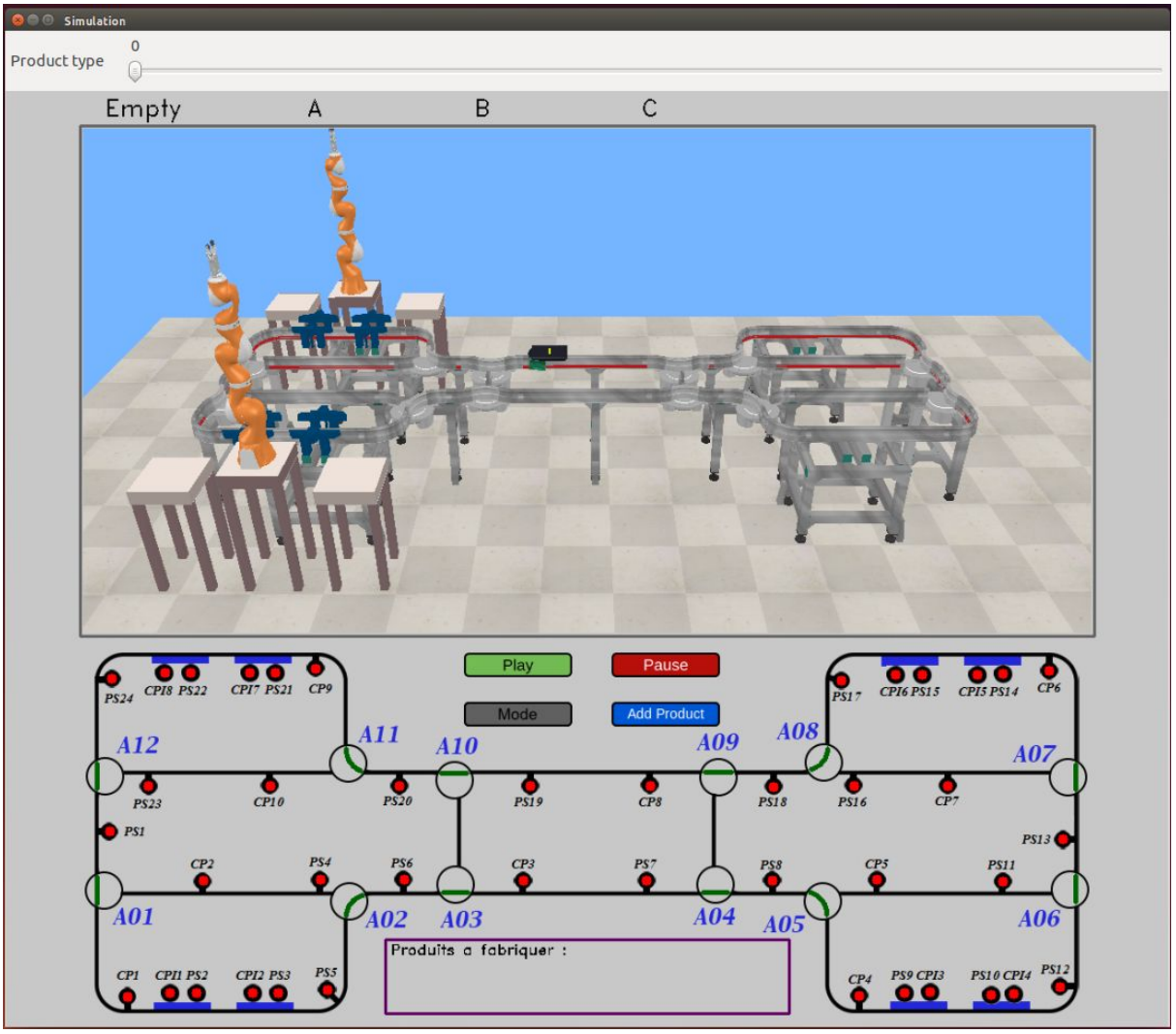

Figure 2 - Nouvelle interface utilisateur

Gestion des aiguillages

À l'origine, la gestion des aiguillages était cachée aux étudiants, en effet, le système connaissant la destination des navettes, les aiguillages étaient programmés afin qu'ils se positionnent pour laisser passer les navettes et leur permettre d'atteindre leur destination. Cependant, pour le TER, la gestion des aiguillages doit être laissée aux étudiants, ce qui n'est pas le cas ici. Ainsi, il faut réaliser une gestion plus fine des aiguillages qui sera intégrée au réseau de Petri que l'étudiant devra mettre en oeuvre.

Ainsi, nous avons supprimé l'intelligence attachée aux aiguillages, puis nous avons ajouté des fonctions couche haute accessibles à l'étudiant permettant de choisir l'orientation d'un aiguillage (droite, gauche …). Nous avons donné l'accès aux différents capteurs (voir figure 2) présents sur la maquette indiquant la présence ou non d'une navette et les actionneurs (devant chaque aiguillage, ou au niveau des postes) permettant de stopper une navette. Toutes ces nouvelles fonctions ont été réalisées en

essayant de garder au maximum ce qui avait été fait les années passées. Ainsi la gestion des aiguillages est entièrement réalisée par les étudiants.

Adaptation de la coloration

Cependant en enlevant l'intelligence mise en place l'année passée, la coloration des navettes et des postes a été altérée et ne fonctionnait plus. En effet pour colorer un élément dans la simulation 3D on a besoin de connaître son "handle" (numéro identifiant cet élément). La solution trouvée est que quand une navette passe un aiguillage ou un poste, on envoie alors le handle de cette navette au prochain point de passage (aiguillage ou poste). La même stratégie a été mise en place les années passées, mais elle s'appuyait sur l'intelligence qui a été supprimée, de plus l'événement qui envoyait le handle au prochain élément était l'activation de l'actionneur qui fait démarrer la navette au niveau de l'aiguillage ou du poste. Cependant synchroniser les envois sur cet événement peut engendrer des bugs non identifiables pour les futurs étudiants. Pour la stratégie adoptée, il a été décidé de choisir comme événement un front descendant sur les capteurs au niveau des aiguillages et postes.

Ensuite des modifications ont dû être apportées aux fonctions permettant de colorer ou décolorer les navettes et postes de travail. En effet on a dû mettre en place l'utilisation du bon handle. Aussi on a simplifié les conditions pour colorer ou décolorer un élément, il suffit maintenant que la main du robot se trouve au dessus de l'élément en question (l'étudiant pourra utiliser des positions prédéfinies pour cela). Il a aussi été également crée une fonction accessible par les étudiants pour savoir si le poste entré en argument a un produit à évacuer.

Aussi des modifications ont été faites dans la gestion de l'avancement d'une tâche. Maintenant l'étudiant doit mettre le produit sur le poste (le poste se colore) et ensuite utiliser une fonction avec en argument la durée d'exécution pour lancer la tâche. Aussi un signal permet maintenant de savoir si sur tel poste une tâche est terminée ou non.

Arrivée et sortie des produits dans le système

Toujours dans le soucis de ressembler le plus possible à la maquette réelle nous avons mis, dans la simulation, l'arrivée des produits sur le poste 2. Nous avons dû faire en sorte qu'un produit n'écrase pas le suivant et qu'il y ait donc une liste d'attente de produits si les navettes n'évacuent pas assez vite de la table.

Les produits peuvent arriver soit en mode manuel, c'est à dire que l'on choisit sur l'interface utilisateur le type de produit et que l'on appuie sur "Add Product" pour

colorer la table (ou pour ajouter une couleur dans la liste d'attente), soit en mode automatique en suivant les instructions du fichier ProductConfiguration.config.

La sortie des produits s'effectue au niveau du capteur CP8 avant l'aiguillage, il a été décidé de mettre l'évacuation des produits à ce niveau-ci car sur le système réel ce tronçon est facilement accessible par les étudiants. Dans la simulation cette évacuation s'illustre par la décoloration des navettes sur front descendant du capteur CP8.

Fichier de configuration des paramètres

Pour faciliter le choix des paramètres de la simulation, un fichier ProductConfiguration.config est laissé libre d'accès aux étudiants et permet de choisir :

- Le nombre de navettes sur le circuit (entre 1 et 6)
- Le nombre de fois que l'on lancera la séquence de produit
- Le temps entre les boucles
- L'écart de temps entre l'arrivée des produits sur une même séquence
- Introduire les gammes des produits (ex: Le produit A doit passer sur le poste 1 puis 2 puis 3)
- Le temps de travail sur chaque poste

```
Start
nbNavettes : 6
nbLoop : 2
deltaLoop : 5
delta : 2 2 2 5
A : 1 3 4 : 1 1 1
B : 1 4 3  : 1 1 1
A : 1 3 4 : 1 1 1
C : 3 : 10
```

Figure 4 - Exemple de paramètres de simulation

Modélisation d'objets en 3D

Une hypothèse forte qui a été prise pour réaliser la simulation est de représenter les produits en colorant les navettes et les postes, plutôt que de les modéliser en 3D. Nous avons donc tenté de travailler sur cet aspect de la simulation. Tout d'abord, afin de réaliser ce travail, il a été nécessaire de monter en compétence sur V-REP. Il est possible et assez simple de faire apparaître un modèle voulu à un endroit voulu, en utilisant le même procédé que pour la génération des navettes, donc la création des produits ne pose pas de problème. On a choisi dans un premier temps de représenter les produits par des cubes, puisque c'est une forme très simple.

La première des choses à tester est le comportement d'un produit lorsqu'il est sur une navette. Lorsque celle-ci se met en mouvement, un premier problème apparaît : le produit "glisse" de la navette, selon l'accélération, le freinage, ou les virages de cette dernière. Une première idée est de jouer sur les propriétés physiques du produit et de la navette : masse, inertie, matériau etc.. Cette première solution n'étant pas tout à fait convaincante, nous devons trouver une autre solution. Pour bien maintenir le produit en place, nous décidons d'intervenir directement sur la modélisation des navettes et d'y ajouter des petites "cales" afin de s'assurer que le produit reste en place tout au long du trajet. Cette solution est meilleure, bien qu'on observe de légers problèmes d'affichage lors du transport (le produit traverse les cales). On peut voir ces cales en bleu sur la figure ci-dessous :

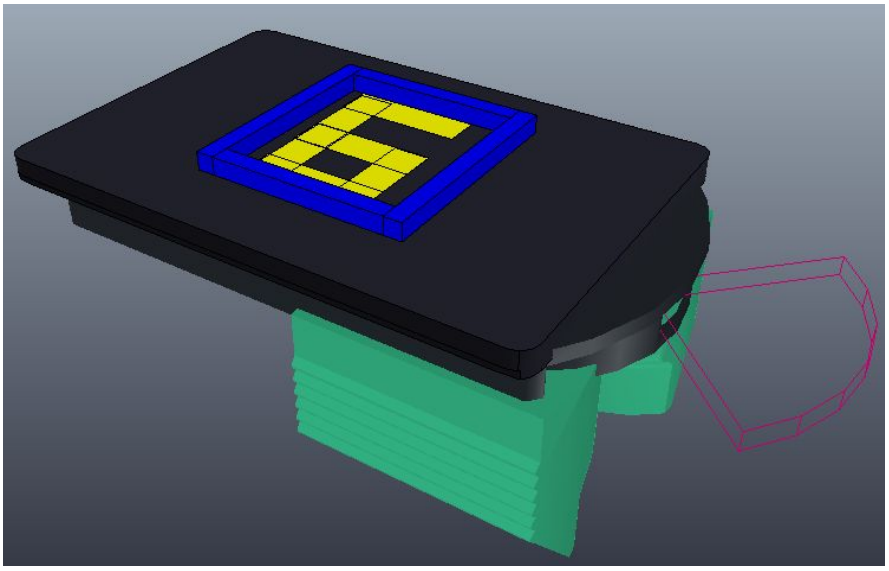

Figure 5 : Modélisation d'une navette sous V-REP

Nous sommes à présent capables de créer un produit sur un plan de travail, ou sur une navette, et de le déplacer *via* les navettes. Nous devons à présent être capables de faire prendre un produit sur une navette et de l'amener à un poste de travail, et vice et versa, en utilisant les robots KUKA à notre disposition dans la simulation. Nous ajustons les coordonnées des positions du robot pour que celles-ci correspondent parfaitement à l'emplacement d'apparition du produit sur la table, et à l'emplacement du produit sur une navette. Nous programmons ensuite notre simulation pour tester la prise du produit par le robot. Le cube que nous avons choisi pour modéliser un produit pose quelques problèmes à la pince de par sa forme, nous décidons donc de créer un produit d'une forme plus simple à attraper :

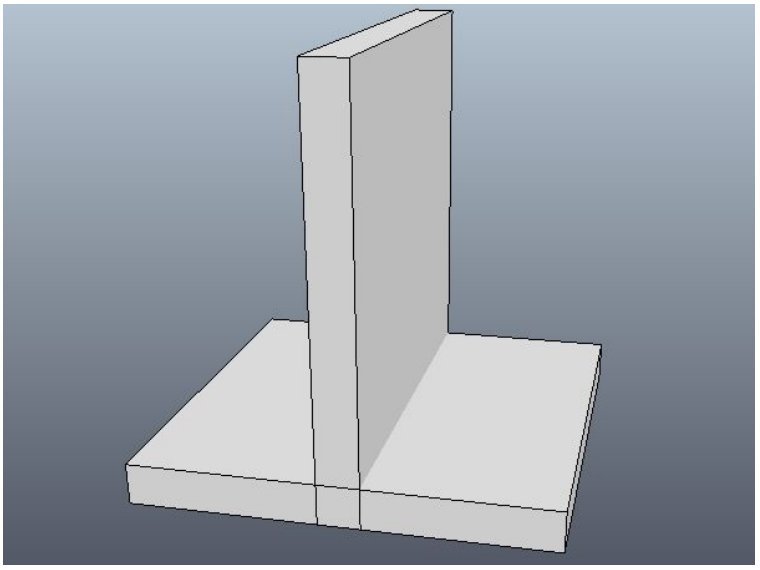

Figure 6 : Modélisation d'un produit sous V-REp

Celle-ci ne pose pas de problème lorsque nous voulons prendre un produit sur un poste et le mettre sur une navette :

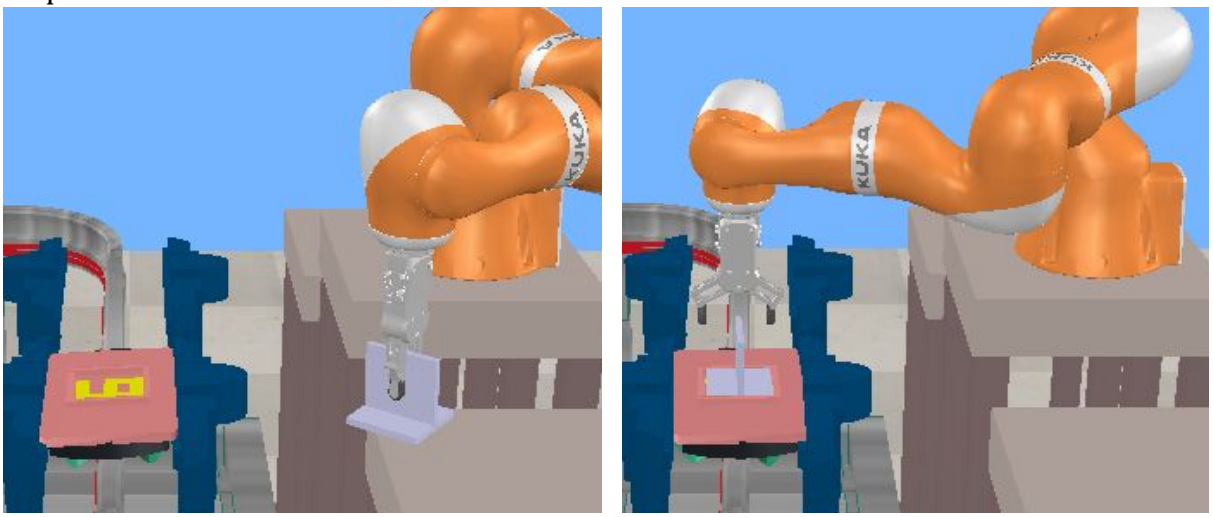

Figure 7 : prise d'un produit par un robot

En revanche, pour des raisons inconnues, le robot prend beaucoup de temps (quelques minutes) pour fermer sa pince lorsqu'il veut prendre une pièce se trouvant sur la navette. De plus, lorsqu'il ferme finalement sa pince pour attraper le produit, il ne parvient pas à le soulever, probablement à cause des cales. À ce jour ce problème n'est pas encore résolu. Une solution peut être de travailler sur le matériau des cales, ou de changer leur forme, ou de trouver une solution alternative aux cales pour fixer le produit à la navette. Une fois cela fait, il faudra configurer les autres postes.

Réalisation du checker

Afin d'aider l'étudiant dans la réalisation de son TER, nous avons décidé de réaliser un checker capable d'informer l'étudiant en cas d'erreur dans son réseau de Petri. Si le fichier ne respecte pas la gamme prévue dans le fichier de configuration ou si le partage de ressources n'est pas respecté par exemple si deux produits sont traités simultanément par le robot.

Avant de réaliser le checker, il faut réaliser un Log file dans lequel on note les évènements importants par lesquels passent le produit. Notamment son traitement sur un poste, son dépôt ou prise sur une navette, l'arrivée et l'évacuation de ce produit ainsi que l'état du robot soit s'il est occupé ou non.

Afin d'optimiser l'utilisation du fichier log par le checker on utilise le modèle d'enregistrement suivant dans le fichier log.

Pour le passage du produit sur un poste :
temps de simulation (teps Vrep) & nom du produit & numéro de poste& d pour début du traitement ou f pour fin de traitement.
Le poste 2 étant le poste d'arrivée des produits, dans ce cas d= date de disponibilité et f la date de prise du produit.

Pour la présence ou non du produit sur une navette :
temps de simulation & nom du produit & 0 (pour indiquer que c'est une navette) & o pour occupée et l pour libérée.

Pour l'état d'un robot :
temps de simulation & numéro du robot & o pour occupé et l pour libre

Malheureusement l'écriture du fichier log n'est pas terminée, nous ne parvenons pas à récupérer les information Vrep dans les main des postes alors que nous incluons le .h du poste correspondant.

Concernant la réalisation du checker, nous avons donc pas pu le réaliser sans le log file, cependant le principe auquel nous avons pensé est de créer deux fonctions gérant respectivement le bon respect des différents contraintes sur le fichier configuration et les contraintes concernant le partage des ressources.
Les contraintes sont les suivantes :

-Respecter la gamme de fabrication inscrites dans le fichier de configuration : l'ordre des produits, le délai entre le lancement de chaque produit, le chemin que doit suivre de

chaque produit et la durée de traitement aux postes par lesquels passe le produit. On vérifie cela en utilisant les dates d'arrivée et de sortie des produits sur les différents postes.

-Vérifier la cohérence du partage de ressources, c'est à dire que chaque ressource ne peut être utilisée que par un produit à la fois. Pour les robots, il faut vérifier que lors de chaque prise de pièce le robot était libre. Pour les navettes ou les postes, il faut vérifier qu'il n'y a avait pas de pièce sur la navette ou sur le poste concerné lors du dépôt d'un produit sur l'un ou l'autre.

Ainsi cela n'a pas encore été réalisé à ce jour, il faut avant tout régler le problème sur le l'écriture du log file.

Mise en place du sujet de TER

Le sujet de TER fait partie des objectifs majeurs de notre projet. Sa rédaction s'est faite en se basant sur le programme pédagogique de la filière GEA. En effet, il a fallu penser à un sujet complet afin que les étudiants puissent mettre en application toutes leurs connaissances des réseaux de pétri, du partage de ressources et de la programmation en langage C++. Le sujet de TER est structuré de façon à simplifier le plus possible la compréhension du système par les étudiants. Nous l'avons structuré de la façon suivante :

Introduction : Une brève introduction afin de présenter les objectifs du TER, préciser les notions qui seront abordées et situer le contexte de l'atelier. Présentation de la plateforme : Comme son nom l'indique, il s'agit de la présentation de la cellule. Elle est identique dans le fond à celle qui a été faite précédemment dans ce rapport (cf Présentation de la cellule flexible).
Le cahier des charges : Il s'agit de la partie la plus importante du sujet. En effet, c'est le cahier des charges qui permet de définir les attentes du TER. Dans ce dernier, on oriente les étudiants sur comment ils devront mener leur projet et quels outils ils utiliseront. L'objectif principal est de réussir à assurer le traitement de plusieurs produits selon leur gamme de production avec un nombre limité de ressources : les robots et les navettes. À l'aide d'un fichier de configuration "ProductConfiguration" qui sera donné aux étudiants, ils devront définir l'ordre et le séquencement du traitement de plusieurs produits.

Ce fichier est assez important dans la mesure où la programmation de toutes les fonctions qui ont permis de réaliser cet atelier a été faite à partir de ce dernier. De plus, il contient les informations permettant d'utiliser correctement les fonctions de haut niveau disponibles. Ainsi comprendre ce fichier est primordial pour les étudiants. Un

autre fichier dédié à la programmation du réseau de pétri en C++ sera également fourni aux étudiants. Pour faciliter la programmation du réseau de pétri et faire correspondre la logique du code et celle des fonctions de haut niveau, l'initialisation a déjà été faite comme on le voit sur la figure ci-dessous :

La réalisation physique de la plateforme n'étant pas encore opérationnelle, le travail se fera uniquement en simulation. Cela représente une information importante puisque les étudiants pourront tester plus souvent leur programme.

La simulation : Nous avons également présenté brièvement la partie opérative de la simulation pour permettre aux étudiants de l'utiliser. En effet, nous avons expliqué tous les fonctions de chaque bouton disponible.

Fonctions disponibles : Pour programmer le réseau de pétri, nous avons mis en place des fonctions de haut niveau. La nécessité de ces fonctions a été abordée par les encadrants pour faciliter le travail des étudiants et se focaliser uniquement sur les connaissances de réseaux de pétri et de partage de ressources. Ces fonctions sont récapitulées dans le tableau suivant :

Objets	Fonctions
Robots	Robots.RobotEnPosition(numéro robot)
	Robots.EnvoyerPosition(numéro robot, position)
Aiguillages	AXX.get_PSXX( ) AXX.get_CPXX( )
	AXX.get_posAig_Droit( ) AXX.get_posAig_Virage( )
	AXX.Droit( ) AXX.Virage( )
	AXX.START( ) AXX.START_VIRAGE( ) AXX.START_DROIT( )
	AXX.STOP( ) AXX.STOP_VIRAGE( ) AXX.STOP_DROIT( )
Commande	cmd.PiecePrise(numéro poste) cmd.PieceDeposee(numéro poste)
	cmd.lancementTache(numéro tâche, temps de réalisation)

	cmd.get_tacheXdone( )
Capteur Poste	Capteur_Poste.StopShuttle(numéro poste) Capteur_Poste.StartShuttle(numéro poste)
	Capteur_Poste.get_PSXX( )

Les XX correspondent à un numéro d'aiguillage ou de capteur, on écrira par exemple A10.STOP( ), A2.STOP( ), A1.STOP( ) etc … (Attention c'est bien A1.STOP( ) et non pas A01.STOP( )). De plus pour pouvoir utiliser les fonctions AXX.get_PSXX il faut écrier le bon AXX, par exemple pour avoir la valeur des capteurs en amont de A12 on écrira A12.get_PS23( ).
Le fonctionnement des fonctions est expliqué en annexe.

Difficultés

Lorsque nous sommes arrivés au laboratoire et que nous avons dû reprendre le projet précédent, nous avons rencontré des difficultés pour nous approprier tous les fichiers et comprendre leur fonctionnement, car même si les étudiants avaient écrit quelques tutoriels, les codes manquaient drastiquement d'explications, et les tutoriels n'étaient pas exhaustifs, nous avons donc essayé de faire au mieux en écrivant des explications. Ce manque de compréhension dû au manque d'explications sur le travail qui a déjà été réalisé a induit de gros obstacles à surmonter par exemple lorsqu'on a rendu modulable le nombre de navettes dans le système (jusqu'à 6) on a utilisé d'autres modèles de navette (la différence étant que le numéro sur la navette est différent) cependant la coloration des navettes ne fonctionnait que sur une navette. Après deux jours de recherche on a découvert que ce bug venait du fait que la définition des modèles dans Vrep est différent de l'unique navette utilisée précédemment. Plus précisément c'était la syntaxe du nom de signal nécessaire pour colorer.

Conclusion

Nous avons travaillé pendant 5 semaines sur ce projet long. Nous avons réalisé avec succès notre premier objectif qui était d'accélérer la simulation. De plus, nous avons retravaillé les fonctions en couche-haute qui ne correspondaient pas aux nécessités d'un TER usine du futur. Elles sont à présent mieux adaptées et plus proche du système réel. Nous avons également travaillé sur le futur de cette simulation : la manipulation d'objets V-REP en 3D.