



SNCF RAPPORT DE STAGE

Étude d'un concept en robotique pour la surveillance des tunnels du RER E

13 juin - 25 août 2022

CONFIDENTIEL

Aéro 4

Julien Fresnel
Assia Belbachir

REMERCIEMENTS

Avant toute chose je souhaite remercier Madame Anne-Lise Combecau Paupière, à l'accompagnement au changement à la Direction Générale IDF et mère d'un ami d'enfance, qui a transmis mon CV en interne au sein de la SNCF et qui m'a permis d'avoir une mise en relation rapide avec plusieurs services de l'entreprise.

Je remercie également Monsieur Louis-Romain Joly, chef de projet de l'équipe robotique et mon maître de stage, pour sa patience, sa gentillesse et son accompagnement durant toute la durée de mon stage. Il aura su me guider tout au long de ma mission en tant que stagiaire et m'aura apporté sa grande expertise et sa bienveillance.

De plus, je remercie toutes les autres personnes de l'équipe robot et humains : Clara Cus-saguet, ingénieure projets innovants et transverses, Enzo Lefèvre, stagiaire, Théo Berillon, stagiaire et Clément Compain, alternant. L'ambiance de travail de l'équipe était très agréable et leur aide et bienveillance était parfaite.

Enfin, je voudrais remercier tous les autres stagiaires du plateau sur lequel je travaillais. L'ambiance globale dans les locaux était vraiment agréable. J'ai eu la chance de rencontrer de merveilleuses personnes, talentueuses et bienveillantes qui ont tout de suite su me mettre en confiance et qui ont rendu mon stage encore meilleur.



Engagement de confidentialité

Toutes les informations recueillies par l'élève stagiaire durant son stage doivent rester confidentielles.

Le devoir de réserve est de rigueur absolue. L'élève stagiaire a l'obligation et prend donc l'engagement de ne jamais utiliser les informations recueillies par lui pour les communiquer à des tiers, sans accord préalable de SNCF et de l'école, y compris le rapport ou mémoire de fin d'études.

Par ailleurs, SNCF peut demander le retrait de certains éléments du rapport ou mémoire estimés par elle confidentiels. Les personnes amenées à en connaître sont contraintes par le devoir de réserve à n'utiliser ni divulguer les informations du rapport ou mémoire.

La diffusion du rapport ou mémoire à la fin du stage, est strictement limitée à trois (3) exemplaires, que ceux-ci soient en version papier ou numérique pour un usage strictement personnel ou interne :

- Un exemplaire pour l'élève stagiaire,
- Un exemplaire pour SNCF,
- Un exemplaire pour l'école.

Toute autre diffusion, totale ou partielle, à titre gratuit ou onéreux, et quel que soit le support, devra recevoir l'accord préalable de SNCF. Ceci vaut tant pour l'élève stagiaire que pour l'école.

En cas d'accord donné par SNCF, elle se réserve toutefois le droit de modifier cette nouvelle diffusion si elle l'estime nécessaire.

L'élève stagiaire s'engage également à ne conserver, emporter, ou prendre copie d'aucun document ou logiciel, de quelque nature que ce soit, appartenant à SNCF, sauf accord écrit de cette dernière.

Cet engagement vaut non seulement pour la durée du stage mais également après son expiration pendant une durée de cinq (5) ans.

Durant le stage et à l'issue de celui-ci, le non-respect de la clause de confidentialité ainsi que le devoir de réserve par l'élève stagiaire constitue un manquement grave. SNCF se réserve, par conséquent, le droit de se rapprocher de l'école afin que cette dernière prenne toute mesure nécessaire.

Enfin, durant le stage comme à l'issue de celui-ci, SNCF se réserve la faculté de faire valoir ses droits en justice à l'encontre de l'élève stagiaire et/ou à l'encontre de l'école, afin d'obtenir réparation de l'ensemble des préjudices qu'elle aurait subis du fait de la violation de la clause de confidentialité.

Fait à Ivry-sur-Seine , le 13 juin 2022

L'élève stagiaire (1),

L'école (1),



(1) Signature, précédée de la mention manuscrite « lu et approuvé »

INTERNE SNCF

NOM ET PRENOM DE L'ELEVE - STUDENT NAME	Julien Fresnel			
NOM TUTEUR/REPRÉSENTANT(S) DE L'ENTREPRISE TUTOR OR COMPANY REPRESENTATIVE	Louis-Romain Joly			
ENTREPRISE (si stage ou contrat) - COMPANY (if work placement or contract)	SNCF			
Compétences mises en œuvre par le stagiaire <i>Competences put in practice by the trainee</i>	Oui Yes	Non No		
A répondu à un besoin opérationnel et/ou fonctionnel par une solution technique <i>Has responded to an operational and/or functional need with a technical solution</i>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
Has conceived or developed a system or equipment	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
A réalisé un équipement ou un système - <i>Has built a system or an equipment</i>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
A conduit ou géré un projet, une étude - <i>Has lead or managed a project or a study</i>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
A eu des responsabilités en management - <i>Has had Management responsibilities</i>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		

note seuil pour valider le stage fixée à : 12/20

Pensez à indiquer Votre NOTE GLOBALE	17	/20
--	----	-----

Merci de cocher la case correspondante

Comportement en situation de travail - Behaviour in the workplace	Insuffisant Insufficient	Moyen Average	Bien Good	Très bien Very Good	N/A Not applied
Ponctualité, Assiduité - <i>Punctuality/ Attendance</i>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Respect des délais par rapport aux travaux demandés - <i>Ability to complete assignments to a deadline</i>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Curiosité intellectuelle - <i>Intellectual Curiosity</i>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Rapidité, Efficacité - <i>Speed and efficiency</i>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Organisation, Méthode, Rigueur - <i>Organisation, Method, Rigour</i>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Motivation et investissement - <i>Motivation</i>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Aptitude à communiquer, sens du contact - <i>Interpersonal Relationships & Communication</i>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Esprit d'initiative - <i>Sense of initiative</i>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Respect des règles de sécurité et du règlement intérieur (si applicable) <i>Respect of Safety Rules (if applicable)</i>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Présentation, aisance - <i>Self Presentation, Confidence</i>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Niveau d'anglais - <i>Level of English</i>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Force de conviction - <i>Persuasiveness</i>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Dynamisme - <i>Dynamism</i>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Sens des responsabilités - <i>Sense of Responsibility</i>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Autonomie - <i>Autonomy</i>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Réalisation des reporting à sa hiérarchie - <i>Achievement of reporting to his superiors</i>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Participation et intégration dans l'équipe ou le service - <i>Team working ability</i>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Compétences Métiers / Vocational Competencies	Insuffisant Insufficient	Moyen Average	Bien Good	Très bien Very Good	N/A Not applied
Niveau technique - <i>Technical level</i>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Propositions d'améliorations techniques - <i>Technical initiative and innovation</i>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Innovation et Créativité - <i>Innovation & Creativity</i>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Utilisation de l'outil informatique (Bureautique et CAO) - <i>Use of IT systems (Office and CAD)</i>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Déetecte les risques - <i>Risk Detection</i>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Esprit d'analyse - <i>Analysis capability</i>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Esprit de synthèse - <i>Ability to synthesize</i>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Connaissances théoriques - <i>Theoretical knowledge</i>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Réussite par rapport aux objectifs - <i>Success in achieving objectives set out</i>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

COMMENTAIRES (obligatoire) - Commentaries (compulsory)

Julien a réalisé un très bon stage. Il sait travailler rapidement et produire des résultats fiables. Julien est autonome et sait quand déclencher au moment opportun les séances de reporting.

	Oui/Yes	Non/No
Le sujet de stage était il lié à la recherche ? - <i>The internship's subject was it related to research</i>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Si oui, le sujet du stage était il en lien avec l'industrie ? <i>If yes, the internship's subject it was related to the industry?</i>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Le mémoire de stage est il confidentiel ? <i>The report is confidential ?</i>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Suivi post stage (AERO 5 uniquement) - Work placement follow up (only ING3)
Aptitude à occuper un poste d'ingénieur (Oui/Non)? <i>Is the trainee able to occupy an engineering position (Yes/No)?</i>
De quelle nature (ex. ingénieur calcul, ingénieur d'affaires ...) ? <i>What type of position (eg. CAD engineer, business engineer) ?</i>
Si non, expliquer les raisons / <i>If not, explain the reasons :</i>

Merci de renvoyer ce document dès que possible / Please return the completed document asap

mail : stage@ipsa.fr

Adresse/ Address : IPSA - 63 Boulevard de Brandebourg 94200 IVRY SUR SEINE

TABLE DES MATIÈRES

REMERCIEMENTS	2
FICHE DE SYNTHÈSE	7
INTRODUCTION	8
LE GROUPE SNCF	9
Son histoire	
L'entreprise aujourd'hui	
Quelques informations clés	
LA DIRECTION TECHNOLOGIES, INNOVATIONS ET PROJETS GROUPE	11
La DTIPG	
Son organisation	
Robots & Humains	
ENVIRONNEMENT DE TRAVAIL	13
Conditions de travail	
Localisation	
Horaires	
Matériel et espace de travail	
Ambiance de travail	
Outils de travail	
Robot Operating System	
Rviz	
Gazebo	
URDF, XACRO et SDF	
Python	
Visual Studio Code	
Méthode SCRUM	
Définition	
SCRUM pendant le stage	
LE PROJET REVOLVE	18
Introduction au projet	
Objectifs du projet	

CONFIDENTIEL

Déroulement de la mission

 Détermination des différents concepts

 Préparation de l'espace de travail ROS

 Modélisation des concepts

 Concept 1

 Concept 2

 Concept 3

 Concept 4

 Modélisation de l'environnement

 Marches de 20 cm

 Marches de 25 cm

 Détermination des données physiques

 Simulations et variations paramétriques

 Concept 1

 Concept 2

 Concept 3

 Concept 4

 Conclusion sur les concepts

Navigation du robot

 Quelle solutions ? Quelle différences ?

 Automatisation

 Robotisation

Bonus

 Création de l'environnement

 Description

 Control

 Gazebo

CONCLUSION	40
BIBLIOGRAPHIE / WEBOGRAPHIE	41
SIGLES ET ABRÉVIATIONS	42
GLOSSAIRE	43
RÉSUMÉ	44
ABSTRACT	44

FICHE DE SYNTHÈSE

Auteur : Fresnel Julien - Aéro 4 - SM

Sujet de stage	Objectifs
Conception d'un robot autonome navigant à travers les tunnels du RER E.	<ul style="list-style-type: none"> - Mettre en pli, (modélisation 3D) le robot suivant les indications du maître de stage - Etablir le modèle du robot pour ROS au format URDF. - Développer le driver ROS du robot en C++ et éventuellement en Python
Client principal	Outils utilisés
SNCF	ROS, Gazebo, Python, Excel, LaTeX
Études réalisées	
<ul style="list-style-type: none"> - Tri des différentes idées de concept de robot. - Modélisation des idées principales et simulations de celles-ci sur Gazebo. - Études des variations paramétriques des concepts possibles. - Étude de la navigation optimale pour le concept sélectionné. 	
Résultats	Explications des écarts possibles
<ul style="list-style-type: none"> - Parmi quatre concepts seulement un seul est exploitable. - La robotisation est le type de navigation le plus pertinent. 	<ul style="list-style-type: none"> - Les valeurs données à notre environnement de simulation n'ont pas forcément de dimension physique. - Les conditions réelles sont suivante différentes des environnements de simulation
Difficultés rencontrées	Travaux à poursuivre
<ul style="list-style-type: none"> - Le projet REVOLVE est un sujet de recherche donc difficultés à s'adapter à la méthode de travail d'un projet de recherche - Difficultés rencontrées sur le logiciel Gazebo qui peut se montrer capricieux - Difficultés sur certains sujets liés à des problématiques mécaniques - Difficultés dans la déductions de certains résultats 	<ul style="list-style-type: none"> - Plan de conception du modèle sélectionné - Conceptualiser le robot - Développer les driver ROS - Réaliser la navigation robotisé du robot - Essais en conditions réelles

INTRODUCTION

Afin de valider sa quatrième année d'étude à l'IPSA, il est obligatoire de réaliser un stage, assistant ingénieur ou technicien, d'une durée minimale de deux mois. Ce stage a pour but de mettre l'étudiant face à un travail technique en entreprise ou en laboratoire universitaire. Ce dernier se retrouve plongé en milieu professionnel, environnement qui lui permet une mise en pratique des ses connaissances acquises durant sa formation, la production d'un travail mesurable et d'approfondir son savoir faire et mener une réflexion sur les enjeux de sa mission ou de son projet.

D'un point de vue personnel, ce stage m'aura permis tout d'abord de découvrir les métiers de la robotique. En effet, c'était un domaine qui avait attiré mon attention lors de mon apprentissage à l'IPSA et je voulais approfondir mes connaissances dedans et de découvrir à quoi pouvait ressembler un rôle d'ingénieur lié à ce domaine. De plus, mon travail étant lié à la recherche, c'était un domaine tout nouveau pour moi et donc un apprentissage supplémentaire.

J'ai débuté mes recherches de stage à partir de la fin d'année 2021 lorsque j'étais encore en semestre international à San Diego. Au départ, j'avais des idées précises de ce que je voulais faire : du cloud, de l'intelligence artificielle et de la cyber-sécurité. Malheureusement, après près d'une centaine de demandes de stage, d'envois de CV et de lettres de motivation je n'avais toujours rien. Au début du mois de mai 2022, j'ai décidé de contacter mon réseau proche en lui demandant de joindre mon CV et ma lettre de motivation à tous ceux qui pourraient être intéressés.

Deux semaines après avoir contacté mon entourage j'ai reçu plusieurs propositions et notamment deux de la SNCF. Une proposition dans le milieu de la data, avec des missions qui semblaient similaires à celles effectuées lors de mon premier stage, et une proposition en robotique. Le projet proposé en robotique était très intéressant et le domaine me permettait de sortir de ma zone de confort, j'ai décidé ainsi d'accepter la proposition de stage en robotique.

La Société nationale des chemins de fer français (**SNCF**) est présente dans plusieurs types de transports ferroviaires tels que : le transport de voyageurs ou le transport de marchandises et elle réalise également toute la gestion et l'exploitation du réseau ferré français.

La compagnie fut créée officiellement le 1er janvier 1938, regroupe plusieurs filiales comme SNCF réseau, SNCF voyageurs ou même Keolis.

Ce grand groupe français est un moteur pour l'innovation et la recherche ferroviaire en France et en Europe et cherche constamment de nouvelles techniques et technologies pour le bien de ses passagers. C'est pourquoi, elle possède son propre pôle de recherche dans lequel figure notamment l'équipe Robot&Humains dans laquelle je travaillais.

Dans un premier temps je vais présenter la SNCF d'une manière détaillée puis le service et l'équipe dans laquelle j'ai eu la chance de travailler. Enfin, je parlerai de ma mission, j'expliquerai et détaillerai les outils que j'ai utilisé et leur intérêt dans le cadre de mon projet au sein de l'entreprise.

LE GROUPE SNCF

Son histoire

La Société Nationale des chemins de fer français est créée en application du décret-loi de 1937, visant à fusionner les cinq grandes compagnies ferroviaires du pays. C'est le 1er janvier 1938 que naît officiellement la **SNCF**.

Peu de temps après, la Seconde guerre mondiale est déclarée et l'Allemagne nazie envahit le pays. Le 22 juin 1940, la convention d'armistice prévoit la mise à disposition "pleine et entière" des chemins de fer français au chef allemand des transports. À partir de 1942, la Wehrmacht Verkehrsdirektion utilise les chemins de fer français pour la déportation des juifs, ce qui suscite un sentiment de rejet auprès des cheminots qui le manifestent par des grèves et des actes de résistances quotidiens.

Au sortir de cette guerre, la SNCF se lance dans un projet de reconstruction avec notamment la remise en l'état et l'électrification du réseau ferré. En 1967, l'entreprise se lance dans un service à grande vitesse avec Le Capitole qui atteint les 200 km/h. C'est seulement à partir de 1974, que le projet de train à grande vitesse se lance. En 1981, le Train à Grande Vitesse (**TGV**) bat un premier record de vitesse avec une pointe à 380 km/h. En 1989, le TGV Atlantique est inauguré et a pour spécification de pouvoir rouler à 300 km/h en service commercial. Les années qui suivent marqueront la mise en service des trains Thalys, de l'Eurostar et des TGV Duplex.

Dans le même temps, les Train Express Régionaux (**TER**) ainsi que les voitures Corail vont être mis en service afin de desservir de la meilleure façon le territoire français.

L'entreprise aujourd'hui

Depuis le 1er janvier 2020, l'entreprise a radicalement revue son organisation. Elle s'organise à présent autour d'une société mère et de cinq sociétés anonymes formées en adéquation avec le nouveau pacte ferroviaire.

Cette société mère, **SNCF SA**, regroupe notamment la direction Technologies, Innovation et Projets de Groupe, le service dans lequel j'ai eu la chance d'effectuer mon stage.

Les cinq autres sociétés sont :

- SNCF Réseau qui s'occupe de gérer, maintenir et développer le réseau ferré national, en donnant la priorité au réseau existant.
- Rail Logistique Europe, anciennement Transport Ferroviaire et Multimodal de Marchandises (**TFMM**), qui gère les activités de fret et de logistique ferroviaire.
- SNCF Voyageurs, qui s'occupe de la mobilité de ses voyageurs en répondant à des besoins en termes d'offre, de coûts, de qualité de service et de respect de l'environnement.
- Geodis, partenaire de croissance de la SNCF. L'entreprise est reconnue pour son expertise et sa maîtrise dans l'ensemble des métiers de la supply chain. Geodis se place parmi les leaders de son domaine dans le monde.
- Keolis, filiale à 70% de la SNCF, c'est le leader mondial du métro automatique et du tramway. Le groupe exploite également des réseaux de bus, cars, trains, transports à la demande.

CONFIDENTIEL

mande , navettes autonomes, vélo, ... pour près de 300 autorités organisatrices de mobilité dans 15 pays.

Les différentes entités sont représentées sur l'organigramme ci-dessous :

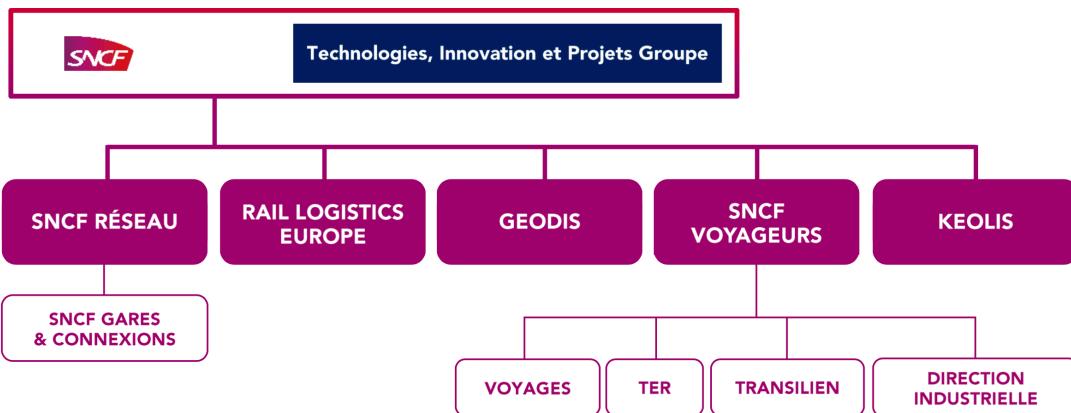


Figure 1 - Organisation du Groupe SNCF

Quelques informations clés

La SNCF est quatrième dans le classement mondial des plus grandes compagnies ferroviaires avec un chiffre d'affaires de **34.8 milliards** d'euros en 2021 malgré la crise COVID. En 2018, l'entreprise investit 9 milliards d'euros dans le matériel roulant et les infrastructures pour le confort des voyageurs. La France est classée dixième dans le monde et deuxième en Europe en terme de longueur du réseau ferroviaire. Cela montre l'importance du ferroviaire dans les transports en France. Le but de la SNCF est d'être un modèle d'excellence et de détermination mondial des services de mobilité et de logistique.

De plus, la SNCF transporte environ **5 millions** de voyageurs par jour, dans 14 000 trains, dont 3.5 millions proviennent des mobilités en Île-de-France. Le réseau ferroviaire français est le deuxième réseau en Europe avec 30 000 km de lignes dont 2 700 km de ligne à grande vitesse.

La SNCF embauche plus de 270 000 collaborateurs dont 60% dans le ferroviaire et 210 000 en France.

Enfin, l'entreprise doit répondre à trois grands enjeux pour réaliser son ambition : la transition écologique, les nouvelles concurrences et les attentes principales, en agissant sur quatre leviers principaux : la maîtrise des coûts, la **digitalisation**, le service client et la décarbonation.

LA DIRECTION TECHNOLOGIES, INNOVATIONS ET PROJETS GROUPE

La DTIPG

La Direction Technologies, Innovations et Projets Groupe (**DTIPG**) accompagne la transformation de l'entreprise. Son but est la recherche dans le domaine du ferroviaire afin de rester à la pointe de l'innovation. Ce service embauche 134 collaborateurs qui travaillent sur environ une vingtaine de projets, tels que **TECH4RAIL**, un programme, qui a pour but, entre autres, de définir le système ferroviaire du futur qui sera autonome, ou encore **Europe's Rail** qui est un programme inédit de recherche collaborative rassemblant l'ensemble des grands acteurs du ferroviaire européen.

La DTIPG en chiffres c'est par exemple :



70 M€
budget en 2022



765
brevets déposés

Son organisation

La DTIPG est dirigée par Carole Desnost et est constituée de huit services différents, parmi lesquels on retrouve notamment la Direction Jeux Olympiques et Paralympiques 2024 ou encore la direction de la Recherche, service dans lequel figure mon équipe. Chaque service est constitué de plusieurs départements, eux-mêmes constitués d'équipes de travail.

Par exemple, l'équipe Robots & Humains dans laquelle j'ai travaillé appartient au Département physique du système ferroviaire qui appartient à la Direction de la Recherche. Voici un organigramme détaillé qui explique comment s'organise cette DTIPG.

CONFIDENTIEL

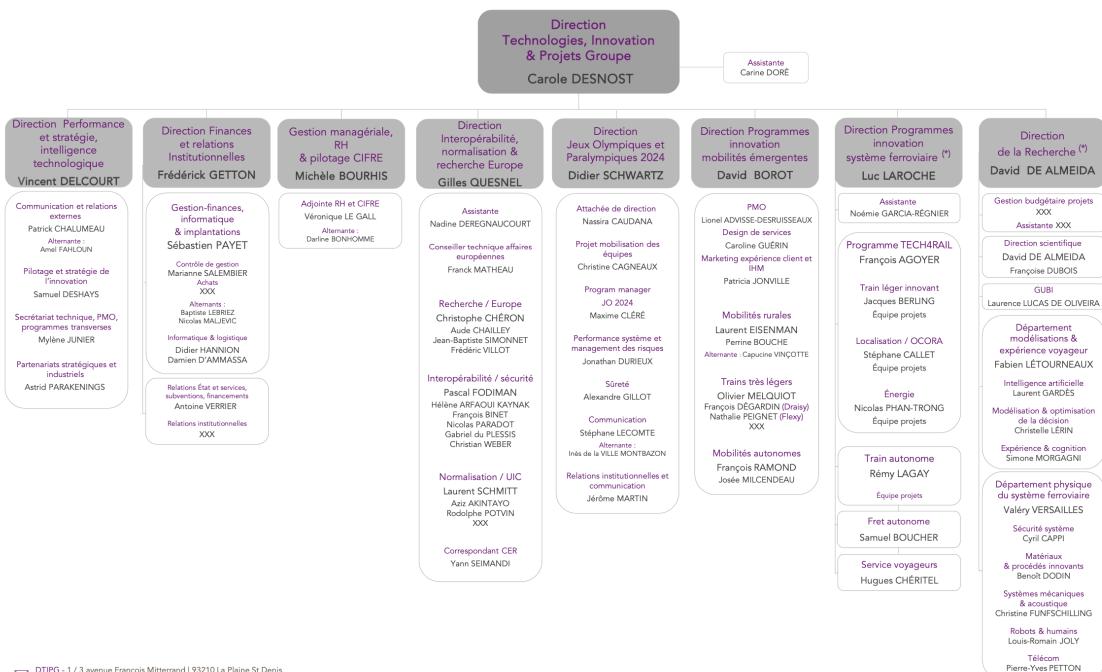


Figure 2 - Organigramme de la DTIPG

Robots & Humains

L'équipe dans laquelle j'ai eu la chance de travailler est l'équipe Robots & Humains. L'objectif de cette équipe est de trouver des solutions innovantes pour d'améliorer plusieurs domaine différents, comme par exemple la surveillance des tunnels ou la maintenance des équipements, à l'aide de robots.

Cette équipe est constituée de quatre personnes à plein temps : Louis-Romain Joly qui est le chef de projet robotique, Clara Cussaguet, ingénierie, Clément Compain, alternant à l'école Polytech Paris-Saclay et Mégane Sartoré, doctorante en psychologie. En plus, de ces quatre personnes, quatre autres stagiaires ont complété l'équipe pendant la durée de mon stage : Enzo Lefèvre, Théo Berillon, Juliette Dubois et moi-même.

Chaque stagiaire a travaillé sur un projet :

- Enzo Lefèvre et Théo Berillon ont travaillé avec Clara Cussaguet sur un projet nommé **NABOT**, acronyme de Navigation Autonome à Bord des Trains, consistant à développer un robot capable de se déplacer de manière autonome à bord des transiliens dans le but d'effectuer certaines tâches spécifiques comme la désinfection des rames et des trains.
- Clément Compain travaillait notamment sur le projet **PICAUTO**, qui est une perche d'inspection avec une caméra à son extrémité et qui a pour but de détecter l'usure des bandes d'archet grâce à un certain algorithme.
- Juliette Dubois, travaillait avec Mégane Sartoré sur un autre aspect de ce projet PICAUTO.
- Quant à moi, j'ai été affecté à un projet nommé **REVOLVE**, qui a pour but de concevoir un robot autonome pouvant parcourir rapidement les tunnels du RER E, lorsqu'une alerte est envoyée, et ainsi identifier la cause de cette dernière. Le but étant de réduire de deux heures à vingt minutes les retards induits suite à un incident.

ENVIRONNEMENT DE TRAVAIL

Conditions de travail

Localisation

Récemment, le groupe SNCF a rassemblé ses filiales et différents services autour du même campus à Saint-Denis, en Seine-Saint-Denis. Le bâtiment dans lequel se situe la DTIPG est au 1-3 avenue François Mitterrand - Immeuble le Jade, à équidistance de la gare Stade de France - Saint-Denis sur le RER D et la gare La plaine - Stade de France sur le RER B. Globalement, sans problème sur les trains, j'avais en moyenne 45 à 50 minutes de transport.

Horaires

Concernant mes horaires de travail, du lundi au jeudi j'arrivais entre 8h30 et 9h et repartais entre 17h et 17h30. Le vendredi en revanche je pouvais n'être présent que la demi-journée de 8h30 à 13h30. Au total, j'avais 35 heures de travail par semaine.

Matériel et espace de travail



Figure 3 - Open Space

Lors de mon arrivée, le premier jour, au sein de la SNCF on m'a tout de suite remis un ordinateur portable sous Windows ainsi que ses équipements, un badge d'accès à l'immeuble ainsi qu'un poste fixe sous Ubuntu.

On m'a également remis mes accès à mon compte SNCF, avec mon adresse email : julien.fresnel@sncf.fr.

Concernant mon espace de travail, le plateau sur lequel tous les stagiaires avaient un bureau était complet, on m'a dans un premier temps installé dans le **Fab'lab**, le labo de l'équipe Robots & Humains. Au fur et à mesure du temps certaines places se sont libérées j'ai donc récupéré un poste de travail au niveau du plateau

avec tous les stagiaires.

Les infrastructures de la SNCF sont toutes neuves et offrent des accommodations agréables pour travailler dans de bonnes conditions. Nous avions des box dans lesquels nous pouvions nous isoler pour passer des appels seuls. Nous avions aussi un espace détente avec un babyfoot où nous passions nos pauses de déjeuner.



Figure 4 - Fab'lab

Ambiance de travail

L'ambiance de travail était excellente entre tous les membres du plateau. Tous les stagiaires étant situés au même endroit, nous pouvions communiquer et créer des liens car nous étions globalement tous de la même génération. En ajoutant à cela les pauses repas et le babyfoot, l'ambiance de travail était d'une très bonne qualité.

Outils de travail

Robot Operating System

Le premier et principal outil que j'ai dû utiliser est **ROS**. Acronyme de Robot Operating System, ROS est un ensemble d'outils informatiques "open source" pour une application dans la robotique. C'est un ensemble d'outils qui nous aide à fabriquer nos propres robots. Le côté open source de ROS, fait de lui un logiciel en évolution continue. C'est un outil créé pour et par la communauté.

Les versions les plus stables de ROS sont sur Linux, bien qu'il existe d'autres versions sur macOS. Il est possible d'avoir d'autres alternatives pour l'utiliser, en passant notamment par des sites Internet comme TheConstruct.

L'idée principale de ROS est de proposer des fonctionnalités standardisées tout en faisant abstraction du matériel. Utiliser ce système d'exploitation comporte certains avantages tels que les nombreux packages disponibles, les fonctionnalités standardisées, les outils de simulation et de visualisation tels que Rviz ou Gazebo ou le fait qu'il soit compatible avec un très grand nombre de robot : motorisés, bras robotisé, ...

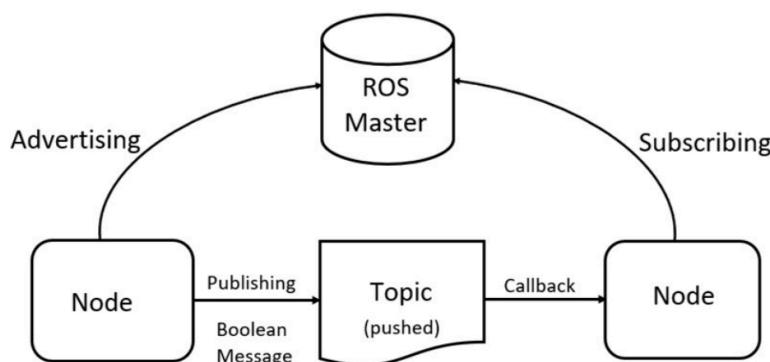


Figure 5 - Représentation de ROS

L'architecture de communication de ROS est plutôt souple. La connexion entre nodes est gérée par le **master**. Et chaque node peut communiquer avec d'autres via des topics. Les **nodes** justement, sont une instance d'un exécutable ou processus, cela peut correspondre à un capteur, par exemple. Les **topics**, permettent d'échanger de l'information entre les différents nodes, cela peut être par exemple une commande de vitesse sur un robot. D'autres éléments de ROS sont importants comme les bags, les messages,... tout cela sera défini plus tard lors de leur utilisation au sein du projet.

C'est ce que l'on appelle le DDS, ce qui constitue qu'une partie du travail de ROS.

Rviz

Rviz est l'outil principal de visualisation 3D de ROS. Il permet entre autres de vérifier si la modélisation de notre robot est bonne à l'aide de nombreux outils, comme une visualisation des axes des liaisons de ce modèle, des parties qui le constituent ou encore, par exemple, la posi-

CONFIDENTIEL

tion et l'orientation relative du robot dans l'espace. Ce sont des outils très importants et qui m'auront été d'une grande aide pendant mon stage.

Gazebo

Gazebo est l'outil de simulation principal associé à **ROS**. La simulation est essentielle lors de la création de systèmes robotisés. Gazebo a l'avantage d'être un outil puissant et rapide. Il permet de réaliser de manière précise et efficace des simulations d'un ou plusieurs robots dans des environnements complexes. C'est un outil complet, puisqu'il est nécessaire d'indiquer les paramètres physiques qui constituent chaque objet de l'environnement que ce soit le décor ou bien le robot. Nous y reviendrons également ultérieurement lors de la description de la mission.

URDF, XACRO et SDF

XML est un langage de balisage extensible, c'est un langage reconnaissable par son usage de chevron (<,>) encadrant le nom des balises.

XML est utilisé pour le format **URDF** (Universal Robot Description Format), permettant de modéliser le robot. Ce sont des fichiers essentiels à ROS pour comprendre et réaliser les simulations. C'est ce qui définit toute l'architecture du robot. L'inconvénient avec ce genre de fichier c'est qu'ils peuvent être très redondants si certaines parties du robot se ressemblent, comme par exemple des roues. C'est là que XACRO intervient.

XACRO est un macro langage basé sur XML qui permet la simplification des fichiers URDF. Il utilise ce que l'on appelle des macro paramétrables, qui correspondent globalement à une forme redondante qui pourra être réutilisé indéfiniment sans devoir la réécrire.

Le Simulation Description Format (**SDF**) est un format XML permettant de décrire des objets et des environnements pour la simulation et la visualisation. Ce format a été à la base développé pour Gazebo, et c'est dans ce contexte que j'ai eu l'occasion de m'en servir.

Python

Python est un langage de programmation interprété, multiparadigmes et multiplateformes. Il favorise la programmation impérative structurée, fonctionnelle et orientée objet. Il est doté d'un typage dynamique fort, d'une gestion automatique de la mémoire avec ramasse-miette (garbage collector) et d'un système de gestion d'exceptions. J'ai eu l'occasion d'utiliser Python lors de la création des scripts de contrôle du robot car c'est un langage parfaitement géré par ROS, j'expliquerai tout cela lors de la description de mes missions.

Visual Studio Code

Visual Studio Code est un éditeur de code source, aussi appelé **IDE** (Integrated Development Environment), fait par Microsoft pour Windows, Linux et macOS. C'est l'outil principal que j'ai utilisé pour écrire mes codes en XML ou Python ainsi que pour naviguer rapidement à travers l'arborescence de mon environnement ROS.

Méthode SCRUM

Le deuxième outil important que j'ai utilisé lors de mon stage est un outil de gestion de projet.

Définition

La méthode SCRUM est une méthode agile de gestion de projet. Elle permet de définir des objectifs à court terme et de fragmenter le projet en objectifs et tâches. Le but est d'améliorer la productivité de l'équipe sur un projet spécifique. Cette méthode fonctionne sur le principe de sprints, c'est-à-dire des cycles de travail d'un mois, durant lesquels des réunions quotidiennes sont organisées et à l'issue desquelles une réunion de fin de sprint est tenue. Les réunions quotidiennes, aussi appelées Daily Scrum, permettent à l'équipe de se coordonner et d'interagir pour la journée. Les réunions de fin de sprint quant à elles sont l'occasion de faire le point sur le travail effectué ainsi que définir les tâches et objectifs du sprint suivant.

Il existe deux rôles majeurs dans la tenue de la méthode SCRUM :

- **Le SCRUM master** : Il est responsable de la tenue des différents évènements SCRUM, il anime les réunions quotidiennes et planifie les réunions de fin de sprint.
- **L'équipe SCRUM** : C'est simplement toutes les personnes travaillant sur le sprint. Je faisais partie de cette équipe.

En plus des deux rôles importants, l'équipe utilise un outil important afin de suivre l'avancée des tâches dans le sprint :

- **Le product backlog** : C'est un tableau récapitulatif de toutes les tâches à réaliser durant le sprint. Il définit également les objectifs du mois. Ce tableau est mis à jour lors des Daily SCRUM pour tenir compte de l'avancée du travail.

SCRUM pendant le stage

Dans le cadre de mon stage, nos réunions de Daily SCRUM avait en réalité lieu tous les deux jours à 8h45, donc le lundi, mercredi et vendredi. Nous informions l'équipe de l'avancée de nos tâches en indiquant leur état : "pas commencée", "en cours", "fait" ou encore "bloquée". Cela permettait d'avoir une visibilité claire sur l'avancée du travail de chacun.

Les réunions de fin de sprint avaient lieu durant toute une matinée et permettaient de faire le point sur le sprint précédent. Lors de celles-ci nous définissions aussi les nouveaux objectifs et tâches du mois et réalisions des démonstrations de notre travail aux autres membres de l'équipe.

Cette méthode était très efficace puisqu'elle nous permettait de se rendre compte de l'avancée de notre travail, de faire le point avec les autres membres de l'équipe et de faire part d'éventuelles difficultés. Elle instaurait aussi un climat de cohésion au sein de l'équipe.

LE PROJET REVOLVE

Introduction au projet

Le projet **REVOLVE** consiste à concevoir un robot permettant de faciliter l'analyse d'incidents dans les nouveaux tunnels du RER E. La difficulté de ce projet est de trouver une solution technique étant capable de rouler proche des rails, sur les passerelles, à une vitesse élevée et également de franchir des marches d'une taille de 20 ou 25 cm. Dans le cadre de la réalisation de cette solution, il m'a fallu d'abord déterminer les meilleurs concepts possible de robot, permettant d'avoir une solution compacte, stable, agile et efficace. Il m'a fallu par la suite, également déterminer le meilleur type de navigation possible pour ce robot, c'est-à-dire automatisé ou robotisé.

Objectifs du projet

Ce projet a des objectifs multiples.

Dans un premier temps, il fallait vérifier que les concepts envisagés soient exploitables dans le contexte d'une montée de marches, contrainte principale du projet.

Ensuite, il a fallu déterminer, pour chacun de ces concepts, les meilleures valeurs paramétriques permettant au robot une meilleure efficacité.

Enfin, l'automatisation ou la robotisation de la montée des marches constituait le dernier objectif de ce projet pendant ma période de stage.

Tous ces objectifs sont en réalité la mission que j'ai dû effectuer tout au long de ma présence au sein de l'équipe **Robots & Humains**.

Déroulement de la mission

Détermination des différents concepts

A mon arrivée, Louis-Romain, mon maître de stage, m'a montré une sélection de schémas de différentes idées de concepts qui avaient été proposées lors d'une première réunion. Ma première mission a été de reprendre ces concepts et de définir leur validité par rapport aux exigences et contraintes définies. Ces idées, schématisées sur papier, étaient nombreuses et je me suis alors servi de mon esprit d'analyse et des outils mathématiques, notamment trigonométriques, pour calculer des longueurs de validité de certains paramètres comme l'empattement ou le rayon des roues par exemple. Ce premier tri sur les concepts a constitué une étape importante pour la suite de la mission puisqu'il nous a permis d'extraire et de définir de nouvelles idées par rapport à celles, réalisables ou non, déjà évoquées.

Au fur et à mesure de ce tri certains concepts étaient complètement supprimés, mais pour d'autres nous n'arrivions pas à savoir avec une feuille et un stylo s'il serait possible de les réaliser. C'est pourquoi, au bout de la deuxième semaine de réflexion, Louis-Romain m'a demandé d'en modéliser certains sur ROS et de réaliser des simulations sous Gazebo pour définir la faisabilité de ces derniers.

Préparation de l'espace de travail ROS

Avant de débuter toute modélisation j'ai dû préparer mon espace de travail ROS sous Linux. A mon arrivée, ROS avait déjà été installé sur mon poste fixe, il fallait alors que je crée mon environnement **catkin_workspace**.

Pour cela, il fallait dans un premier temps exécuter la commande suivante dans le terminal de Linux, qui permet de sourcer l'environnement.:

```
source /opt/ros/noetic/setup.bash
```

Ensuite j'ai dû créer et construire un catkin workspace, c'est un fichier dans lequel je peux modifier, construire et installer des packages. Un package ROS est ce qui peut contenir les nodes, des library ROS, un dataset, des fichiers de configuration ou n'importe quoi qui constitue un module utile. Le but de ces packages est de fournir des fonctionnalités utiles et une manière simple de "consommer" afin que le logiciel puisse être réutilisé. Pour construire ce catkin workspace il a fallu que j'exécute les commandes suivantes dans l'ordre :

```
mkdir -p ~/catkin_ws/src
cd ~/catkin_ws/
catkin_make
```

Après avoir exécuté toutes ces commandes, mon espace de travail était enfin prêt pour que je commence les modélisations des concepts. Enfin, pas tout à fait. En réalité, j'ai également dû organiser mon espace de travail et créer mes packages, je l'expliquerai en partie **bonus**.

Cependant, une chose est importante à noter, à chaque démarrage de mon poste fixe, il fallait que j'exécute des commandes spécifiques pour "activer" mon environnement de travail. Pour gagner du temps j'ai demandé au service informatique de regrouper ces commandes sous l'alias : **sncf_catkinws**

Modélisation des concepts

Nous avons donc extrait quatre potentiels concepts de robot qui étaient à essayer en environnement de simulation. Ces modélisations **URDF** ont été faites grâce à des fichiers **XACRO**.

Concept 1

Le premier concept envisagé avait des bras rotatifs à l'avant et à l'arrière du robot, ces derniers lui permettaient de prendre appui sur les marches avant et arrière. Le but principal était de surélever le robot pour lui faire franchir les marches plus simplement. C'est le premier concept que j'ai dû réaliser. J'ai alors créé mon fichier XACRO de zéro en définissant les formes de chaque partie du robot (chassis, roues, bras rotatifs, ...), j'ai dû définir aussi les balises **<collision>**, qui déterminent la physique de chaque partie du robot, ainsi que les balises **<joints>**, qui permettent de définir les types de liaisons (pivot, rotule, ...).

Voici une partie du code que j'ai dû réaliser.

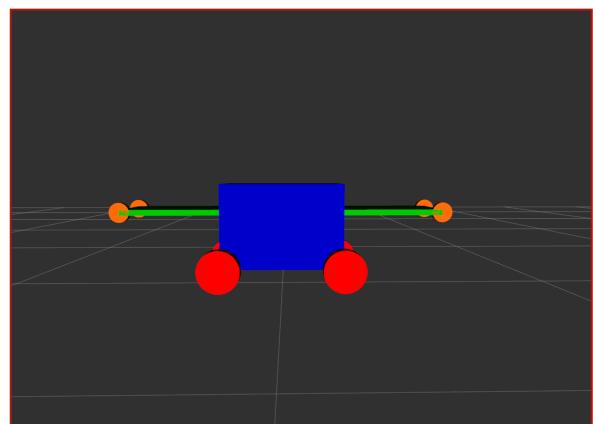


Figure 6 - Concept 1

```

○○○
<!-- MAIN WHEEL: MACRO -->

<xacro:macro name = "main_wheel" params='position side'>
  <link name = "${position}_${side}_wheel">
    <visual>
      <xacro:wheel_origin position = '${position}' side = '${side}' />
      <geometry>
        <cylinder length = "${wheel_length}" radius="${main_wheel_radius}" />
      </geometry>
    </visual>

    <collision>
      <xacro:wheel_origin position = '${position}' side = '${side}' />
      <geometry>
        <cylinder length = "${wheel_length}" radius="${main_wheel_radius}" />
      </geometry>
    </collision>
    <xacro:cylinder_inertial mass ="${wheel_mass}" radius ="${main_wheel_radius}" length
    ="${wheel_length}" />
  </link>

  <joint name = "${position}_${side}_main_wheel_joint" type="continuous">
    <xacro:wheel_origin joint position = "${position}" side ="${side}" />
    <parent link="${position}_wheel_axe"/>
    <child link = "${position}_${side}_wheel"/>
    <axis xyz = "0 1 0"/>
  </joint>
</xacro:macro>

```

Code 1 - Macro roue principale concept 1

Concept 2

Ce concept 2 est une variante du premier, la seule différence réside dans le fait que les bras rotatifs avant et arrière peuvent aussi se déployer dans le sens longitudinal. Le but des bras est alors de surélever le robot avant de franchir les marches, ils prennent appui sur le sol, avant de se déployer et de lever les roues du robot. L'angle de rotation autorisé sur les bras sert juste à ajuster leur position au niveau du sol pour permettre une montée des marches optimale.

```

○○○
<!-- SLIDING ARM: MACRO -->

<xacro:macro name = 'sliding_arm' params='position'>
  <link name = "${position}_sliding_arm">
    <visual>
      <xacro:sliding_arm_origin position="${position}" />
      <geometry>
        <box size="${arm_length} ${arm_width} ${arm_height}" />
      </geometry>
      <material name="yellow"/>
    </visual>
    <collision>
      <xacro:sliding_arm_origin position="${position}" />
      <geometry>
        <box size="${arm_length} ${arm_width} ${arm_height}" />
      </geometry>
    </collision>
    <xacro:box_inertial mass = "${arm_mass}" width="${arm_length}" height="${arm_height}"
    depth="${arm_width}" />
  </link>

  <xacro:if value="${position=='front'}">
    <joint name="${position}_sliding_arm_joint" type ="prismatic">
      <xacro:sliding_arm_origin joint position = "${position}" />
      <parent link="${position}_rotary_axe"/>
      <child link = "${position}_sliding_arm"/>
      <axis xyz ="1 0 0"/>
      <limit effort="1000" velocity="0.2" lower="0" upper="${sliding_arm_length}" />
    </joint>
  </xacro:if>

  <xacro:if value="${position=='rear'}">
    <joint name="${position}_sliding_arm_joint" type ="prismatic">
      <xacro:sliding_arm_origin joint position = "${position}" />
      <parent link="${position}_rotary_axe"/>
      <child link = "${position}_sliding_arm"/>
      <axis xyz ="1 0 0"/>
      <limit effort="1000" velocity="0.2" lower="0" upper="${-sliding_arm_length}" />
    </joint>
  </xacro:if>
</xacro:macro>

```

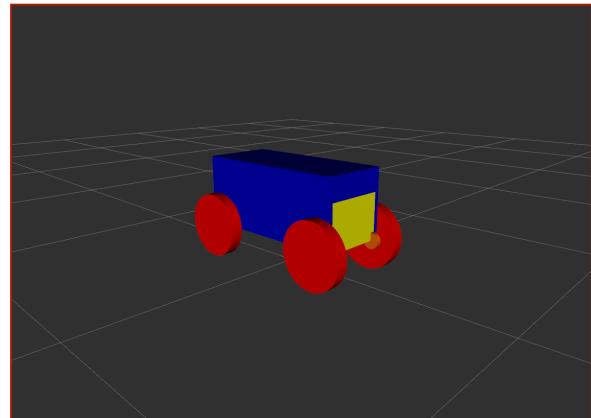


Figure 7 - Concept 2

Code 2 - Macro bras

Concept 3

Cette troisième idée reprend les deux idées précédentes en les mixant. Les roues principales sont directement positionnées sur les bras rotatifs, il y a moins de roues que sur les deux solutions précédentes et nous avons aussi, à la différence des autres modèles, rendu le châssis "solide", c'est-à-dire qu'une balise `<collision>` a été ajoutée, pour que la caisse repose sur les arrêtes des marches lors de la montée.

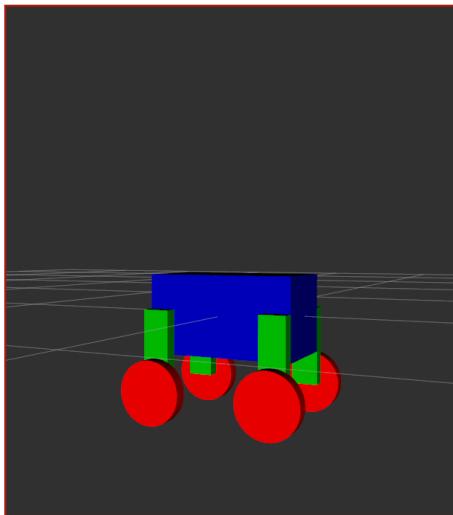


Figure 8 - Concept 3

```
○ ○ ○

<!-- base frame -->

<link name="base_link">
    <visual>
        <acro:default_origin/>
        <geometry>
            <box size="${wheelbase} ${robot_base_depth} ${robot_base_height}" />
        </geometry>
        <material name = "blue"/>
    </visual>
    <collision>
        <acro:default_origin/>
        <geometry>
            <box size="${wheelbase} ${robot_base_depth} ${robot_base_height}" />
        </geometry>
    </collision>
</link>

<joint name="dummy_link" type="fixed">
    <parent link="base_link"/>
    <child link="dummy_base_link"/>
</joint>
```

Code 3 - base_link concept 3

Concept 4

Enfin ce quatrième et dernier concept change radicalement des trois premiers. En effet, nous avons décidé de reprendre une idée souvent exposée, celle des roues de "caddie". Nous avons alors six roues de chaque côté du robot ainsi qu'un "triangle" de trois bras au bout duquel sont positionnées les roues. C'est le concept qui pourrait être le plus difficile à concevoir, compte tenu de la forme de ses roues. Il faudrait réussir à réaliser une transmission mettant en rotation les "triangle" mais également les roues "rouges". Tout cela de manière simultanée ou alternée. Et il est très difficile de faire cela avec la forme de notre robot.

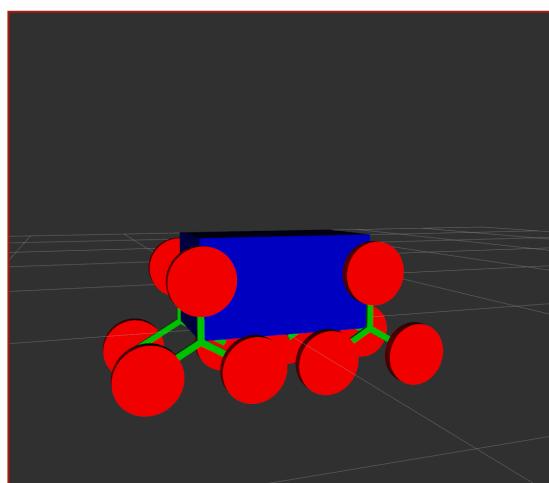


Figure 9 - Concept 4

Code 4 - Joins des roues concept

Modélisation de l'environnement

Après avoir modélisé les quatre concepts de robot, j'avais besoin d'un environnement de simulation pour réaliser mes essais. Il m'a alors fallu réaliser cet environnement pour **GAZEBO**. Nous avions besoin d'un environnement réaliste qui respecte les contraintes présentent dans le tunnel du RER E. La contrainte principale concerne la forme et la taille des marches. Il faut savoir que dans le tunnel du RER E, au niveau des passerelles, il existe deux types de marches différentes, des marches d'une hauteur de vingt centimètres et des marches d'une hauteur de vingt-cinq. J'ai alors modélisé les deux types de marches pour réaliser les essais dans toutes les situations possibles.

Marches de 20 cm

Pour les deux types de marches j'ai réalisé deux versions. Une version avec des contremarches, qui facilite la modélisation et les premiers essais et une version sans ces contremarches qui est la version qui se rapproche le plus de la réalité. A l'aide des données qui m'étaient fournies, j'ai pu réaliser les calculs de certaines longueurs et ainsi la modélisation dans un fichier **SDF**, qui est le format de fichier adapté à Gazebo.

Voici à quoi ressemblent ces marches de vingt centimètres :

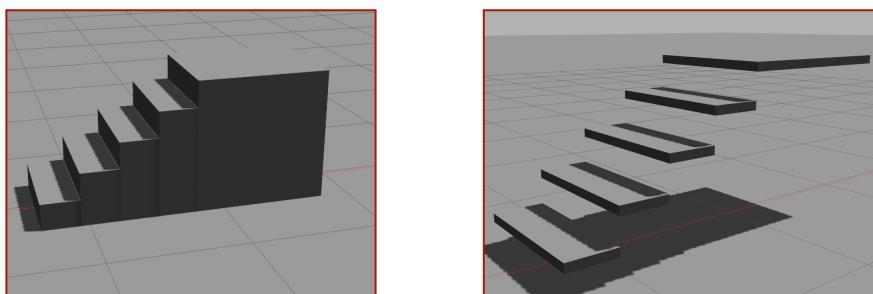


Figure 10 - Marches de 20 cm avec (à gauche) et sans (à droite) contremarches

Marches de 25 cm

Pour ce type de marches, c'est le même principe, cependant au lieu d'avoir cinq marches nous n'en avons que quatre. J'ai également réalisé deux versions différentes.

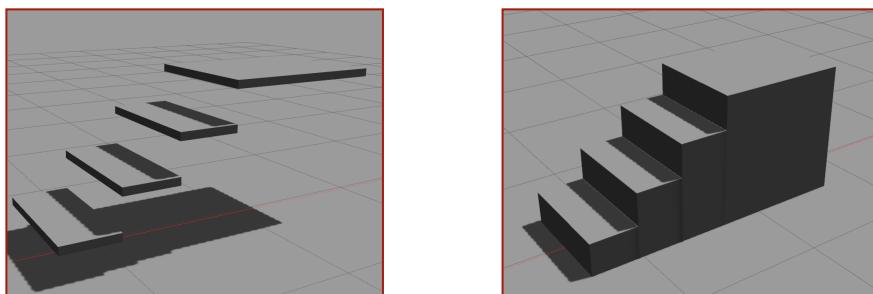


Figure 11 - Marches de 25 cm avec (à droite) et sans (à gauche) contremarches

Détermination des données physiques

Comme je l'ai expliqué au début de ce rapport, l'outil Gazebo est utilisé pour des simulations robotiques. Cependant, pour les effectuer dans un environnement, il nous faut définir des données physiques permettant au logiciel de calculer les efforts, les contraintes, ... et toutes les caractéristiques physiques que l'on retrouve dans le monde réel.

Les données physiques peuvent s'appliquer au **robot** ou à **l'environnement**. Elles se définissent lors de la construction du monde pour l'environnement et dans un fichier **.gazebo** en ce qui concerne celles du robot. Ce fichier est également écrit en XML et nécessite d'importer un certain package de Gazebo.

Que ce soit pour l'environnement ou le robot, on retrouve globalement les mêmes types de balises permettant de définir : la raideur, l'amortissement, les frictions, le nombre de contacts maximum, ... Seules quelques balises sont exclusives à l'environnement comme la balise **<ode>** par exemple, qui permet de définir le type de solveur utilisé pour le calcul des contacts ou encore la balise **<gravity>** qui, comme on peut le déduire, permet d'indiquer si les objets sont sous l'influence de la gravité.

Il existe un certain nombre de balises et de données physiques définissables. Pour savoir lesquelles me seraient utiles, j'ai consulté la documentation Gazebo et SDF sur Internet.

Il faut savoir aussi que certaines valeurs associées à une caractéristique vont au-delà d'une représentation physique. Cette modélisation du contact est simple voir simpliste. Il est donc très difficile de disserter sur la valeur des paramètres que nous renseignons. Les coefficients de frottement, sont eux important (dix fois l'ordre de grandeur réel).

```
<xacro:macro name = "main_wheel_gazebo" params = "position side">
  <gazebo reference = "${position}_${side}_wheel">
    <material>Gazebo/Red</material>
    <mu1>2.0</mu1>
    <mu2>1.0</mu2>
    <kp>10000000</kp>
    <kd>1.0</kd>
    <fdir1>0 0 1</fdir1>
    <max_contacts>10</max_contacts>
    <min_depth>0.0</min_depth>
  </gazebo>
</xacro:macro>
```

Code 5 - Exemple de paramétrisation physique du robot

Simulations et variations paramétriques

Maintenant que tous nos éléments de simulation sont modélisés, nous pouvons commencer les simulations et définir les variations paramétriques possibles de chaque concept. Pour chacun d'entre eux, nous avons réalisé un script de contrôle sous **Python** qui permet de publier des commandes sur les **topics** du robot et ainsi de le contrôler à l'aide des touches du clavier.

Concept 1

Code

Comme je l'ai expliqué précédemment, j'ai commencé par réaliser un script **Python** qui permet de contrôler le robot avec les touches du clavier. Pour réaliser ce code il fallait déterminer les topics sur lesquels publier le message. Pour ce concept, nous avions donc deux topics importants : **/mobile_base_controller/cmd_vel** et **/arm_effort_controller/command**. Le premier topic permet de publier des commandes de vitesse au robot et ainsi le faire avancer, reculer ou s'arrêter. Le deuxième quant à lui permet d'indiquer la position des bras (en radians). On peut montrer l'exemple de code suivant.

```
○ ○ ○

def control_robot(self):
    screen = pygame.display.set_mode((100, 100))
    print("function's running")
    cmd_vel = Twist()
    cmd_arm = Float64MultiArray()
    cmd_arm.data = [0, 0]

    running=True
    while running:

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                running=False

            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_KP8:
                    print("setting linear vel to 2")
                    cmd_vel.linear.x = 2

            elif event.key == pygame.K_KP2:
                print("setting linear vel to -2")
                cmd_vel.linear.x = -2

            elif event.key == pygame.K_SPACE:
                print("break")
                cmd_vel.linear.x = 0

            elif event.key == pygame.K_KP6:
                print("setting +0.05 rad to the both arms")
                cmd_arm.data[0] += 0.05
                cmd_arm.data[1] += 0.0
```

Résultats de simulations

Code 6 - Code de contrôle du concept 1

Avant de débuter les tests, j'ai émis des hypothèses pour définir une première valeur de certains paramètres. J'ai alors pensé à une situation où les deux roues étaient en butée sur les marches avant ou arrière. J'ai alors pu réaliser des calculs grâce aux outils trigonométriques et j'ai pu trouver qu'un empattement de 34.9 cm serait nécessaire. J'ai ensuite fait varier le rayon des roues principales, des petites roues au bout des bras, ainsi que la longueur des bras. J'ai obtenu des premiers résultats qui n'étaient pas exploitables dans l'environnement des marches de 20 cm. Voici quelques exemples de résultats obtenus :

Rayon principal (cm)	Petit rayon (cm)	Longueur bras (cm)	Empattement (cm)	Achieved
15	6	28	34.9	FALSE
14	6	28	34.9	FALSE
13	4	28	34.9	FALSE
12	3.5	29	34.9	FALSE

Sur une quinzaine de tests aucun n'était concluant pour cette valeur d'empattement. J'ai décidé alors de choisir des valeurs d'empattement plus grandes pour essayer de garder la roue

CONFIDENTIEL

arrière en contact avec le sol le plus longtemps possible. Cependant, après également de nombreux tests, je n'ai trouvé aucune solution adaptée.

En fait, lors de la montée, un problème surgissait. En effet, il y avait un moment où aucune des deux roues motrices n'était en contact avec le sol. Le robot se retrouvait alors bloqué dans une position particulière soit parce que le bras arrière empêchait l'avancement soit parce que le bras avant surélevait trop le robot. Pour que ce concept soit exploitable, il aurait fallu motoriser également les petites roues au bout des bras, mais cela aurait demandé trop d'actionneurs. Nous avons alors rapidement décidé de ne pas continuer avec ce modèle et de le considérer comme **non-concluant**.

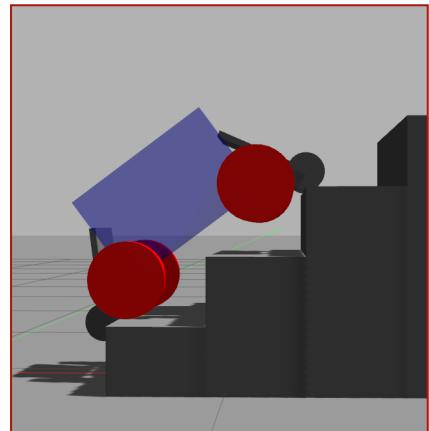


Figure 12 - Situation de blocage concept 1

Concept 2

Code

A l'instar du concept 1, j'ai réalisé un premier script **Python** afin de contrôler le robot avec les touches du clavier. Pour ce code, je devais créer un **Publisher** pour trois topics différents, puisque j'avais trois mouvements à contrôler : la vitesse, le déploiement des bras et la rotation des bras. Les topics en question sont : `/mobile_base_controller/cmd_vel`, `/sliding_arm_position_controller/command` et `/rotary_arm_position_controller/command`.

Les commandes publiées pour définir le déploiement des bras sont exprimées en centimètres et celle pour la rotation en radians. Voici l'exemple de création des publishers :

```
○ ○ ○  
  
class Robot:  
    def __init__(self, robot_name):  
        self.robot_name = robot_name  
  
        self.cmd_vel_pub = rospy.Publisher("/mobile_base_controller/cmd_vel", Twist, queue_size=1)  
        self.cmd_sliding_arm_pub = rospy.Publisher("/sliding_arm_position_controller/command",  
Float64MultiArray, queue_size=1)  
        self.cmd_rotary_arm_pub = rospy.Publisher("/rotary_arm_position_controller/command", Float64MultiArray,  
queue_size=1)
```

Code 7 - Création des publishers concept 2

Résultats des simulations

Dans un premier temps, j'ai appliqué la même méthode que précédemment, c'est-à-dire que j'ai défini une hypothèse de montée pour ce concept en rapport avec ses caractéristiques. J'ai choisi au départ de partir sur les mêmes hypothèses, avec le même choix de longueur. Cependant, j'ai vite déduit que ce ne serait pas possible de faire pareil avec ce modèle. J'ai choisi alors de faire monter deux marches à la roue avant et ensuite faire monter la roue arrière. Il fallait trouver les meilleures valeurs d'empattement possible pour réaliser cela. J'ai obtenu ces résultats :

Rayon principal (cm)	Petit rayon (cm)	Déploiement (cm)	Hauteur entre les axes (cm)	Distance entre les axes (cm)	Empattement (cm)	Achieved
11	3.5	40	20	10	50	FALSE
11	3.5	40	20	10	48	FALSE
11	3.5	40	20	10	52	TRUE

On remarque alors que lorsque l'empattement mesure 52 cm nous avons un concept qui pourrait fonctionner. J'ai donc réalisé des variations sur les paramètres avec cette valeur d'empattement, dans le but de trouver des bornes à chaque donnée.

J'ai réalisé une grande série de tests, approximativement une trentaine, et j'ai obtenu des résultats qui fonctionnaient pour les marches de 20 cm. Malgré quelques difficultés, notamment liées à des bugs de Gazebo qui nous empêchaient d'effectuer des tests sur certaines valeurs, j'ai réussi à obtenir quelques versions de ce concept compatibles avec les marches de 20 cm.

Afin de valider ces résultats, il fallait aussi essayer ces versions sur les marches de 25 cm. Cependant, de toutes les versions obtenues précédemment aucune ne fonctionnait sur le deuxième environnement de marches. J'ai alors revu mon approche et décidé d'adopter la même logique pour les tests mais en commençant sur les marches de 25 cm. Il est bon de noter que le choix des coefficients de frottement pour ce concept est très important, puisque ces derniers déterminent la qualité "d'accrochage" de la roue sur la marche. Ici, les coefficients sont élevés, ce qui implique d'avoir des roues avec une bonne accroche lors de la conception.

Finalement, j'ai obtenu des résultats intéressants pour les marches de 25 cm, ces résultats nécessitent une deuxième validation avec peut-être d'autres valeurs de frottements. J'ai ainsi, testé les modèles sur les marches de 20 cm. Certaines versions étaient valides avec cet environnement. Voici les résultats obtenus avec ce modèle :

CONFIDENTIEL

Rayon principal (cm)	Petit rayon (cm)	Déploiement (cm)	Hauteur entre les axes (cm)	Distance entre les axes (cm)	Empattement (cm)	Achieved
10	3.5	40	20	12	62	FALSE
10	3.5	40	20	14	55	TRUE
14	3.5	40	20	16	55	TRUE
12	3.5	40	20	18	55	TRUE
14	3.5	40	20	20	56	TRUE
14	3.5	40	20	22	57	TRUE
12	3.5	40	20	22	57	TRUE
14	3.5	40	20	22	60	FALSE

Notre empattement est à présent borné en réutilisant les valeurs des tests précédents, ce qui constitue la valeur principale de notre robot. Concernant les hauteurs, distances et tailles des roues, nous utiliserons ce que nous avons trouvé lors des tests. Il est bon de noter que pour les roues, plus elles seront petites, plus la stabilité sera faible, tandis qu'une roue plus grande permettra plus de stabilité mais une plus grande consommation. Concernant la taille de déploiement, elle pourra être définie par l'étude des **bags**.

Un bag est un type de fichier dans ROS permettant de stocker des données, ici en l'occurrence les messages et données publiées sur les topics. Il existe ensuite des outils pour visualiser ces données sous forme de courbes, comme par exemple **plotjuggler**.

Pour enregistrer un bags on exécute dans un autre terminal la commande suivante :

rosbag record -a

Après avoir enregistré les bags, on peut les étudier dans plotjuggler en visualisant les courbes. Cela m'aura été utile pour déterminer l'angle maximum du bras ainsi que la distance maximale de déploiement.

Au final, j'ai trouvé que notre bras devait osciller entre -45° et 45° et que le déploiement du bras devrait faire maximum 36 cm.

Concert 3

Code

On continue la même logique que pour les concepts précédents : avant d'effectuer les simulations dans GAZEBO, j'ai d'abord réalisé un code permettant de contrôler le robot avec les touches du clavier. Cette fois-ci, comme le modèle possède quatre bras. Chaque bras étant indépendant, nous aurons donc un topic par bras en plus du topic pour le diff drive controller. Au total, j'ai créé des publishers pour les cinq topics suivants : `/mobile_base_controller/cmd_vel`, `/front_right_rotary_arm_position_controller/command`, `/front_left_rotary_arm_position_controller/command`, `/rear_right_rotary_arm_position_controller/command` et `/rear_left_rotary_arm_position_controller/command`.

CONFIDENTIEL

Voici un extrait du code qui m'a permis de réaliser le contrôle du robot :

```
○ ○ ○  
class Robot:  
    def __init__(self, robot_name):  
        self.robot_name = robot_name  
  
        self.cmd_vel_pub = rospy.Publisher("/mobile_base_controller/cmd_vel", Twist, queue_size=1)  
  
        self.cmd_front_right_pub = rospy.Publisher("/front_right_rotary_arm_position_controller/command",  
Float64, queue_size=1)  
        self.cmd_front_left_pub = rospy.Publisher("/front_left_rotary_arm_position_controller/command",  
Float64, queue_size=1)  
        self.cmd_rear_right_pub = rospy.Publisher("/rear_right_rotary_arm_position_controller/command",  
Float64, queue_size=1)  
        self.cmd_rear_left_pub = rospy.Publisher("/rear_left_rotary_arm_position_controller/command", Float64,  
queue_size=1)
```

Code 8 - Création des publishers concept 3

Résultats des simulations

Comme d'habitude, j'ai défini mes hypothèses pour la montée qui m'ont permis aussi de déterminer le comportement du robot, c'est-à-dire quand il faut bouger tel ou tel bras et quand je dois le faire avancer. Ce concept, censé être une amélioration des concepts 1 et 2, s'est en fait montrer plus compliqué que prévu.

En effet, lors de la montée et de la rotation de certains bras, il fallait tout le temps faire attention que le centre de gravité soit bien positionné sinon notre robot basculait vers l'arrière ou sur le côté. La difficulté la plus flagrante était lorsque le robot se trouvait sur une marche, que la caisse repose dessus ou non, ce dernier glissait vers l'arrière quasiment systématiquement à cause de la position de son centre de gravité. Après de multiples essais, cette solution a fini par être abandonnée. Il est trop compliqué de trouver une formule viable dans laquelle le centre de gravité de n'est pas trop haut, ni trop bas, ni trop devant ou ni trop derrière et qui permet de franchir sans encombre les marches avec cette proposition de robot.

J'ai donc déterminé ce concept comme **non-concluant**.

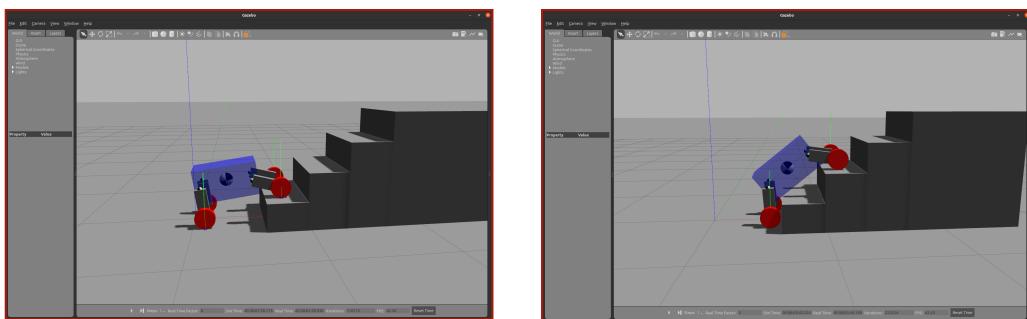


Figure 13 - Situation de difficultés concept 3

Concept 4

Code

Le code de ce concept-là était le plus simple à réaliser. En effet, j'avais juste besoin de contrôler la vitesse du robot ainsi que la rotation des roues en forme de triangles. Il me fallait donc publier des commandes sur seulement deux **topics**. De plus, les quatre "triangles" doivent tourner en même temps donc cela m'a permis de ne pas assigner plusieurs touches pour contrôler indépendamment chaque bras. Au final, je n'avais que quatre touches de clavier assignées ce qui rendait beaucoup plus facile pour moi les simulations dans l'environnement **GAZEBO**.

```

if event.type == pygame.KEYDOWN:
    if event.key == pygame.K_KP8:
        print("setting linear vel to 2")
        cmd_vel_wheel.linear.x = 2

elif event.key == pygame.K_KP2:
    print("setting linear vel to -2")
    cmd_vel_wheel.linear.x = -2

elif event.key == pygame.K_SPACE:
    print("break")
    cmd_vel_wheel.linear.x = 0

elif event.key == pygame.K_KP9:
    cmd_vel_arm.data[0] += 0.05
    cmd_vel_arm.data[1] += 0.05
    cmd_vel_arm.data[2] += 0.05
    cmd_vel_arm.data[3] += 0.05

```

Résultats des simulations

Code 9 - Exemple du contrôle du concept 4

Il est bon d'être vigilant sur certains points essentiels concernant ce concept. Tout d'abord, la présence des triangles peut générer des problèmes de taille au niveau du robot : plus le bras du triangle sera grand, plus la longueur maximale du robot sera élevée. Et dans notre situation il ne fallait pas dépasser les 90 cm de long, sinon le concept pourrait ne pas être exploitable.

Concernant le fonctionnement des simulations, il reste inchangé par rapport aux autres. J'ai cherché à définir une plage de validité des paramètres afin d'avoir la meilleure solution possible. J'ai alors défini des hypothèses de taille sur l'empattement et j'ai effectué des tests et des variations à partir de cela. En faisant varier l'empattement, la longueur des bras des triangles varie également, ce qui est logique puisque ces deux longueurs sont liées.

Au total, j'ai effectué 75 essais, en commençant par les marches de 25 cm. Parmi ces tests, seulement 29 ont été concluants. J'ai alors testé ces 29 solutions concluantes sur les marches de 20 cm pour savoir quelles versions étaient compatibles avec les deux environnements. Sur ces 29 tests, seulement 4 versions ont été retenues, les voici :

Rayon principal (cm)	Taille des bras (cm)	Empattement (cm)	Longueur totale (cm)
9	15	46	89.98
9	15	45	88.98
9	14.5	44	87.11
9	13.5	43	85.25

Le but principal était pour l'instant d'obtenir des valeurs d'empattement fonctionnelles. On remarque aussi que pour les quatre solutions la longueur totale est inférieure à 90 cm.

Nous avons ensuite continué les tests sur les marches sans contremarche, pour faire une première validation, puis nous avons tenté de faire varier les autres paramètres pour chaque valeur d'empattement.

Finalement, nous avons pu extraire de ces tests 7 versions finales complètement valides. Ces versions peuvent être visibles dans le tableau ci-après.

CONFIDENTIEL

Rayon principal (cm)	Taille des bras (cm)	44	45	46
9	13	Non	Non	Non
	14	Non	Oui	Non
	15	Non	Oui	Oui
	16	Non	Non	Oui
	17	Non	Non	Non
10	13	Non	Non	Non
	14	Non	Oui	Oui
	15	Non	Non	Oui
	16	Non	Non	Non
	17	Non	Non	Non
11	13	Non	Non	Non
	14	Non	Non	Non
	15	Non	Non	Non
	16	Non	Non	Non
	17	Non	Non	Non

Ici, le rayon minimal exploitable est de 9 cm. Cependant, une incertitude persiste pour des rayons inférieurs. En effet, cela pourrait fonctionner avec des roues de 8 ou 7 cm de rayon, cependant lors des simulations, j'avais certains bugs avec Gazebo pour ces grandeurs, ce qui m'empêchait de mener à bien les essais.

Finalement, avec ces résultats, nous pouvons d'ores et déjà construire un modèle fonctionnel en utilisant les paramètres définis au-dessus.

Conclusion sur les concepts

Bien que deux concepts sur quatre sont parvenus à effectuer la mission de monter les marches, nous avons décidé de garder pour la suite uniquement le **concept 4**.

En effet, le concept 2 est très intéressant. Cependant, certaines incertitudes persistent concernant l'efficacité du robot. Il faudrait refaire des tests avec des valeurs de friction différentes pour s'assurer que ce modèle est toujours exploitable. Il n'est pas impossible de réaliser ce robot mais concernant les légères incertitudes qui l'entourent, il pourrait ne pas être fiable.

Concernant les concepts 1 et 3, comme nous l'avons expliqué, ils ne sont tout simplement pas concluant. Ce ne serait pas pertinent de les concevoir à moins de vouloir faire un robot voué à l'échec.

Finalement, le concept 4 apparaît comme l'unique concurrent. Il a une agilité remarquable puisqu'il peut monter les marches entre 30 et 45 secondes, ce qui est très rapide. Les versions sélectionnées sont relativement compactes et respectent largement le cahier des charges.

La seule vraie difficulté concernera la conception : Comment réaliser la transmission des roues au bout des bras ? Quelle type de transmission utiliser ?

Toujours est-il qu'il est à ce jour le **meilleur concept**.

Navigation du robot

Quelle solutions ? Quelle différences ?

Avant de débuter cette navigation il a fallu penser à comment la réaliser. Il existe deux types de navigation, soit automatisée soit robotisée. Mais laquelle choisir ?

Automatisée

Dans le cadre d'une navigation automatisée, nous n'avons aucun capteur, nous avons accès uniquement aux données connues : distances, angles, ...

Dans notre cas, nous aurons besoins des données de position du robot à l'instant t ainsi que de la position angulaire de ses triangles.

Pour vérifier si cette automatisation est possible, nous allons étudier les **bags** enregistrés grâce au logiciel **plotjuggler**.

Robotisée

La navigation robotisée quant à elle, utilise des données récoltées par des capteurs, comme par exemple **LiDAR**. Ces derniers permettent alors un comportement plus "intelligent" et efficace du robot puisqu'il pourrait s'adapter à chaque situation. Cependant, les coûts de fabrication seraient augmentés en choisissant cette solution, c'est pourquoi dans un premier temps nous allons essayer d'automatiser la montée.

Automatisation

Machine d'états

Pour réaliser cette automatisation de la montée des marches, j'ai préalablement étudié les **bags**, que j'avais enregistré lors des essais des variations paramétriques afin de prendre connaissance de toutes les données qui nous seront utiles. Ces données sont : certaines distances, des angles de rotation, l'**odométrie** du robot ainsi que sa vitesse.

Maintenant que toutes les données ont été extraites, nous avons pu penser en fonction de celles-ci, à comment réaliser cette automatisation. C'est là qu'intervient le principe de machine d'états, c'est ce modèle mathématique qui me permettra d'établir le comportement de mon robot. Le principe est simple, notre machine d'états (Finite-State Machine en anglais) peut passer d'un état à un autre en réponse à une entrée, c'est ce que l'on appelle transition.

Ici notre **FSM** va être composée de trois états. On peut retrouver la représentation en diagramme UML de cette FSM ci-contre.

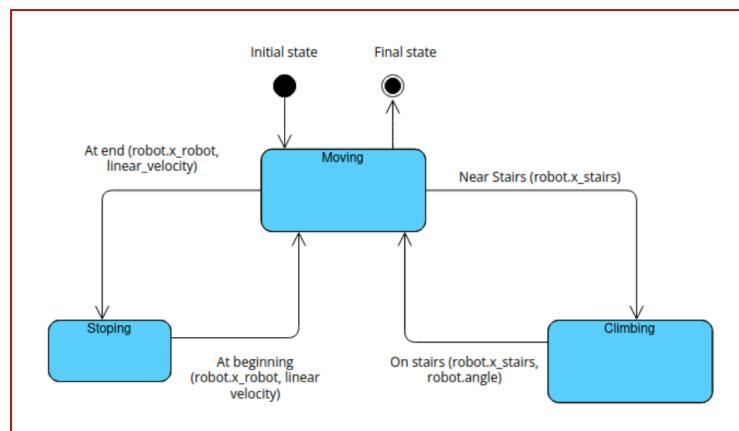


Figure 14 - FSM de notre automatisation

CONFIDENTIEL

Nous avons donc trois états qui permettent de définir les trois actions de notre robot:

- **Moving**, le robot avance
- **Climbing**, le robot monte les marches
- **Stopping**, le robot s'arrête

Cette FSM, et ces trois états, correspondent donc au comportement final de notre robot et permettront de réaliser notre automatisation.

Étude des bags

Avant de faire toute la navigation, j'ai donc étudié les bags. De cette étude (voir courbe ci-contre), j'ai conclu que l'odométrie n'était pas utilisable, ce qui rend l'automatisation impossible, j'y reviendrai plus tard. En effet, les roues tournent selon leur axe mais également selon l'axe de rotation des triangles. Cela fausse toute notre odométrie et fait sens. Néanmoins, le robot ne peut donc pas se repérer dans son espace en se fiant à l'odométrie des roues.

Pour tout de même réussir une automatisation, avant la robotisation ultérieure, j'ai réussi à trouver le **topic** qui permet de récupérer de manière exacte la position du chassis dans l'environnement **Gazebo**. J'ai donc étudié les types de messages publiés sur ce topic et essayé d'extraire les informations les plus importantes de ces derniers. J'ai également extrait la courbe d'évolution des positions de nos triangles. J'ai pu ensuite m'attarder sur le code.

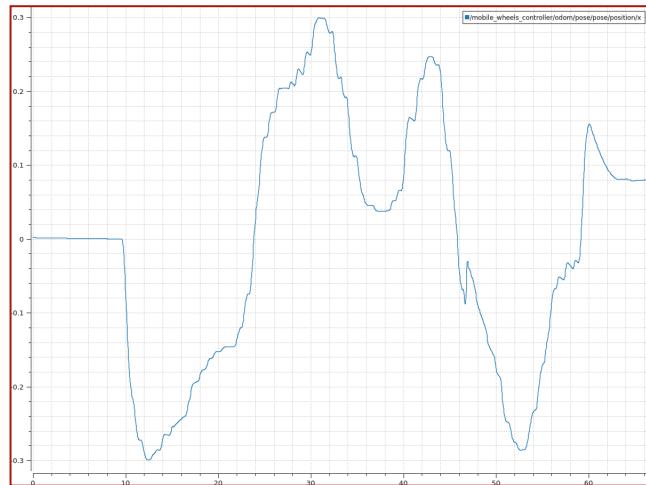


Figure 15 - Odométrie du robot

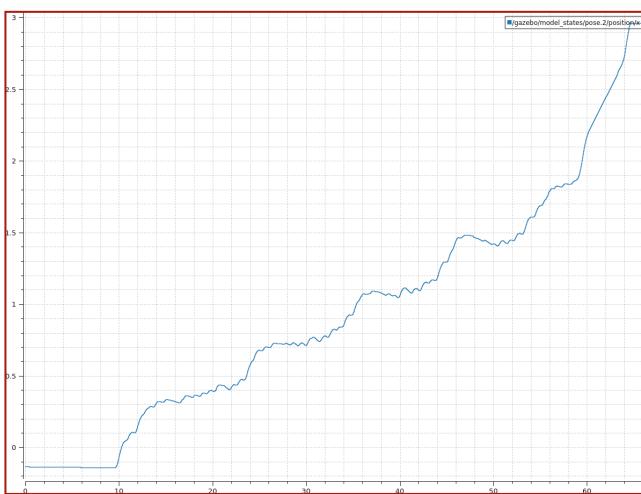


Figure 16 - Courbe selon d'avancement selon x de base_link

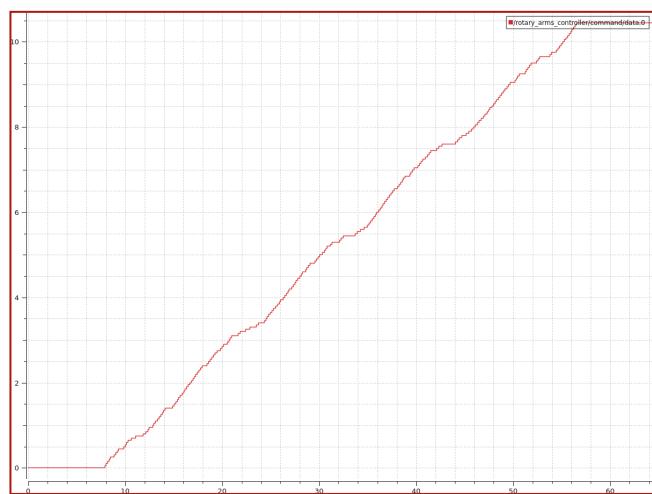


Figure 17 - Courbe d'évolution de la rotation des bras

Programmation de la navigation

D'abord, il faut savoir que, l'environnement comportant deux types de marches, il était très compliqué de faire une solution généralisée. J'ai alors réalisé deux codes, un pour les marches de 25 cm et un autre pour celle de 20, cela montre une autre limite de l'automatisation, j'y reviendrai.

J'ai alors créée une **class Robot** dans lequel j'ai pu assigner des attributs (qui correspondent à des caractéristiques) et des méthodes (permettant de définir un comportement).

Ainsi, dans notre constructeur, nous aurons défini les positions (x,y,z) de notre robot. Ce ne sont que des initialisations, puisqu'on les modifiera en fonction de ce que l'on récupère des topics.

Nous avons aussi deux callback functions. Ces deux fonctions : **callbackpose** et **callbackbasepose** permettent de récupérer respectivement l'odométrie des roues et la position réel du chassis dans l'environnement **Gazebo**. Ensuite, nous avons d'autres fonctions facilement déductibles avec **set_heading_target** et **get_distance**. Enfin, les deux méthodes qui permettent de publier les messages sur les topics : **set_speed_angle** et **set_rotationnal_arm**.

Avec tout ce que nous avons, la class robot créée, notre visualisation de notre FSM et notre étude de bags, nous pouvons créer notre stratégie dans la fonction **run_control**. Cette navigation est pour les marches de 25 cm.

```

○ ○ ○

if (i > 5):
    print("stopping")
    linear_velocity = 0
    robot.set_speed_angle(linear_velocity)
    break

if (position[5] - 0.02 < robot.x_base < position[5] + 0.02):
    print("stopping")
    linear_velocity = 0
    robot.set_speed_angle(linear_velocity, 0)
    break

if (position[i] + 0.02 > robot.x_base):
    print(f"State : Moving")
    linear_velocity = 0.1
    robot.set_speed_angle(linear_velocity, 0)

if (position[i]-0.02< robot.x_base < position[i] + 0.02) and (i<5):
    print(f"State : Climbing")
    angular_position = angles[i]
    robot.angle = angles[i]
    linear_velocity = 0.1
    robot.set_rotationnal_arm(angular_position)
    robot.set_speed_angle(linear_velocity, 0)
    i+=1

```

Code 10 - Exemple de la machine d'état du robot

Les listes **positions** et **angles** sont nos "points de passages" de notre robot. C'est ce qui permet d'indiquer les distances à atteindre et les positions des bras à avoir.

Ce qu'on retient finalement de cette automatisation, c'est que la montée est bien plus rapide puisqu'on franchit les marches en seulement **29 secondes**.

CONFIDENTIEL

Concernant l'environnement des marches de 20 cm, tout sera identique. Seules les listes positions et angles vont être changées, les "points de passages" seront plus fréquents puisque la montée est un peu plus décomposée. Voici les nouvelles listes de positons et d'angles :

○ ○ ○

```
position = [0.2, 0.50593, 0.55593, 0.68593, 0.778, 0.925, 1.05, 1.1, 1.28, 1.4, 1.47, 1.57,  
1.66, 1.8, 2.3]  
angles = [2.6, 3.3, 3.6, 4.4, 5.4, 6.5, 7., 8., 8.97, 9.5, 10.1, 10.9, 11.45, 12.6]
```

Code 11 - Listes des données de positions et d'angles pour la navigation à 20 cm

Finalement, avec l'automatisation pour les marches à 20 cm, le robot franchit la difficulté en **32 secondes**.

Limites de cette solution

Comme je l'ai expliqué précédemment, cette solution n'a en réalité presque que des limites. En effet, dans ce que nous avons réalisé, nous n'utilisons pas l'odométrie des roues. En fait, notre robot se repère dans l'espace parce que les données que nous avons sont fournies par **Gazebo**, ce qui explique que le code fonctionne. Cependant, en situation réelle c'est impossible, il nous faut utiliser l'odométrie locale du robot pour lui permettre de comprendre son espace et qu'il sache où est-ce qu'il est. De plus, la non généralisation du code est aussi un point noir. En effet, il nous faut un robot qui s'adapte à son environnement. Toutes ces conditions rendent notre solution d'automatisation obsolète.

Robotisation

Le choix d'une robotisation est donc essentielle et implique fatalement une installation de capteurs sur le robot.

Capteurs

L'installation de capteurs permettra de rendre notre robot intelligent, de se repérer dans son environnement et de savoir comment réagir et comment se comporter dans cet environnement. Il existe plusieurs types de capteurs. Cependant, ceux le plus fréquemment utilisé dans ces situations et dans la réalisation de projets de robotique sont les capteurs **LiDAR**. Dans notre cas, un LiDAR 2D sera suffisant. Ces capteurs émettent des rayons laser qui rebondissent sur les obstacles et reviennent au capteur. En connaissant la vitesse de la lumière, cela permet de connaître la distance de l'obstacle par rapport au robot mais aussi sa forme. Cela permet aussi d'obtenir une odométrie plus précise de notre robot. Il devient également intelligent. En effet, s'il se décale à cause d'un angle de lacet positif (ou négatif) il pourra alors corriger sa trajectoire et son allure. Notre robot pourra alors complètement connaître son environnement et ainsi nous pourrons créer une navigation plus robuste.

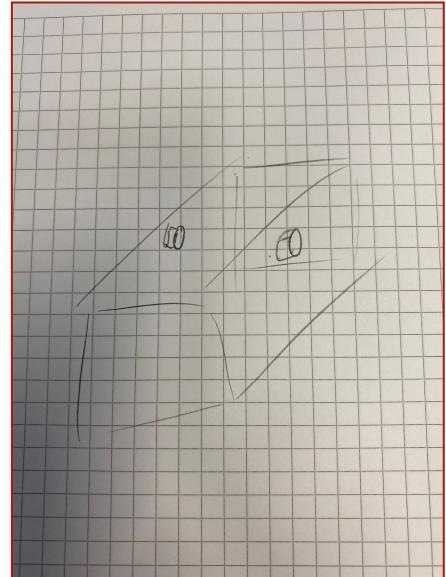


Figure 18 - Positionnement des LiDAR

Finite-State Machine

En ajoutant des capteurs à notre robot nous pouvons complexifier son comportement et ainsi la machine d'état qui permettra de représenter ce dernier. Nous pourrions alors ajouter une correction de lacet ou de vitesse. Voici, ici, un exemple de diagramme UML de cette FSM :

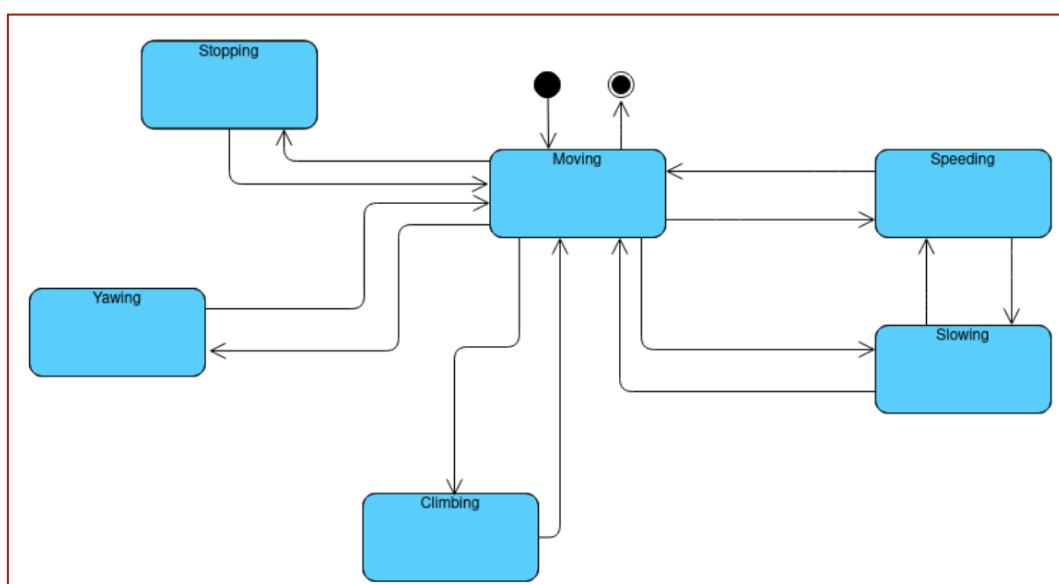


Figure 19 - FSM de robotisation

Programmation de la navigation

Le script **Python** ne sera pas plus compliqué que celui déjà réalisé lors de l'automatisation. Il pourrait cependant être plus général ; la présence de capteurs supprimera les listes positions et angles et nous définirons les conditions en fonction de la distance du robot par rapport aux marches, ce qui va faciliter grandement l'écriture des conditions de notre FSM.

De plus, on pourra se servir des données odométriques fournies par les **LiDAR** et ainsi indiquer des corrections au robot notamment concernant sa vitesse ou son lacet, ce qui lui permettra d'adapter son comportement en fonction des difficultés qu'il rencontre sur son chemin.

Malheureusement, par manque de temps, je n'ai pas pu réaliser toute la partie robotisation. Ce seront des choses effectuées plus tard par l'équipe **Robots & Humains**.

Bonus

Création de l'environnement

Pour parler plus en détail de la création de l'environnement, il a effectivement fallu taper les commandes indiquées plus tôt dans le rapport, mais ce n'était pas suffisant. Il fallait, comme je l'ai dit, créer aussi des **packages**. Les packages contiennent des nodes, libraries, datasets, fichiers de configurations,... Ils se créent grâce à la commande : `catkin_create_pkg`

J'ai donc créé trois packages permettant d'organiser mon travail :

`sncf_robot_descriptpion`, qui est le package dans lequel j'avais tous mes fichiers en lien avec la modélisation URDF des concepts

`sncf_robot_control`, qui est le package permettant de gérer tous les aspects contrôle du robot. Les controllers, les scripts python,...

`sncf_robot_gazebo`, ce package était celui en rapport avec GAZEBO et l'environnement de simulation, les modélisations des marches,...

Description

Pour parler plus en détail de ce package : il y avait donc dans un dossier urdf, tous mes fichiers **XACRO** de description des concepts.

Il y avait aussi un dossier rviz, qui permettait d'enregistrer les configurations **RVIZ** de notre robot. C'est ce qui nous permettait de visualiser notre modélisation sans ouvrir tout l'environnement de simulation.

Enfin, il y avait un dossier launch, dans lequel étaient stockés les fichiers `display.launch` de chaque concept. Ces fichiers permettent de créer les nodes RVIZ pour chaque concept et donc d'afficher la modélisation.

Control

Ce package était, je pense, le plus fourni et le plus important. En effet, c'est dans ce package qu'il y avait, dans un dossier script, tous les scripts Python que j'ai réalisé et qui permettaient de contrôler avec le clavier le robot ou juste la navigation autonome.

Mais ce qui constituait les fichiers les plus importants étaient ceux dans le dossier control. Ces fichiers `.yaml` permettent d'indiquer le type de controller de notre robot. Par exemple, pour faire avancer un robot à roues, il nous faut un **diff drive controller**. C'est dans ces fichiers `.yaml` que l'on définissait ce type de controller.

Enfin, il y avait aussi des fichiers launch.

```

mobile_wheels_controller:
  type: "diff_drive_controller/DiffDriveController"
  left_wheel:
    ["front_left_1_main_wheel_joint","front_left_2_main_wheel_joint","front_left_3_main_wheel_joint",
     "rear_left_1_main_wheel_joint","rear_left_2_main_wheel_joint","rear_left_3_main_wheel_joint"]
  right_wheel:
    ["front_right_1_main_wheel_joint","front_right_2_main_wheel_joint","front_right_3_main_wheel_join
     t",
     "rear_right_1_main_wheel_joint","rear_right_2_main_wheel_joint","rear_right_3_main_wheel_joint"]
  publish_rate: 50
  pose_covariance_diagonal : [0.001, 0.001, 1000000.0, 1000000.0, 1000000.0, 1000.0]
  twist_covariance_diagonal: [0.001, 0.001, 1000000.0, 1000000.0, 1000000.0, 1000.0]
  #wheel_separation: 0.3
  #wheel_radius: 0.15
  wheel_separation_multiplier: 1
  wheel_radius_multiplier: 1
  cmd_vel_timeout: 0.25
  enable_odom_tf: false
  base_frame_id: base_link
  linear:
    x:
      has_velocity_limits : true
      max_velocity       : 8.0 # m/s
      min_velocity       : -1 # m/s
      has_acceleration_limits: true
      max_acceleration   : 0.8 # m/s^2
      min_acceleration   : -0.4 # m/s^2
      has_jerk_limits    : true
      max_jerk           : 5.0 # m/s^3
    angular:
      z:
        has_velocity_limits : true
        max_velocity       : 1.7 # rad/s
        has_acceleration_limits: true
        max_acceleration   : 1.5 # rad/s^2
        has_jerk_limits    : true
        max_jerk           : 2.5 # rad/s^3

```

Code 12 - Exemple de fichier `.yaml`

Gazebo

Enfin, comme je l'ai expliqué, dans ce package étaient présentes toutes les modélisations de l'environnement. Les marches avec et sans contremarches ainsi que certains autres environnements de tests réalisés.

Il y avait aussi dans ce package les fichiers `simu.launch`, de chaque concept qui permettaient de lancer leur simulation sur **Gazebo**.

Dans ce fichier, on chargeait la modélisation URDF, les controller, et on indiquait l'environnement à utiliser. On pouvait ajouter quelques paramètres de position du robot au début de la simulation.

```
○ ○ ○

<launch>
  <arg name="world" default="stairs_4"/>

  <include file="$(find gazebo_ros)/launch/empty_world.launch">
    <arg name="world_name" default="$(find sncf_robot_gazebo)/worlds/$(arg world).world"/>
    <arg name="paused" value="false"/>
    <arg name="use_sim_time" value="true"/>
    <arg name="gui" value="true"/>
    <arg name="headless" value="false"/>
    <arg name="debug" value="false"/>
  </include>

  <arg name="robot_namespace" default="/" />
  <arg name="x" default="0.0"/>
  <arg name="y" default="0.0"/>
  <arg name="z" default="0.4"/>
  <arg name="yaw" default="0.08"/>

  <include file="$(find sncf_robot_description)/launch/display_4.launch"/>
  <include file="$(find sncf_robot_control)/launch/control_4.launch"/>

  <node name="spawn_model" pkg="gazebo_ros" type="spawn_model"
    args="-x $(arg x)
          -y $(arg y)
          -z $(arg z)
          -Y $(arg yaw)
          -urdf
          -param robot_description
          -model robot" />
</launch>
```

Code 13 - Exemple de fichier `.launch`

CONCLUSION

Pendant ce stage, j'ai été amené à sortir de ma zone de confort. En effet, effectuer un stage dans le domaine de la robotique, c'est choisir un domaine dans lequel j'ai encore beaucoup à apprendre, comprenant des outils techniques et académiques que je ne maîtrise pas pleinement. C'est ce qui a rendu ce stage si enrichissant d'un point de vue professionnel. J'ai appris énormément de nouvelles choses, que ce soit par rapport à **ROS**, à des domaines de mécaniques ou bien même simplement par rapport à la **SNCF**, son fonctionnement ainsi que tout le milieu ferroviaire.

Mon stage, guidé par une unique mission de recherche, m'a poussé à aller plus loin dans mes réflexions, voir toujours les problèmes en amont et les anticiper, comme par exemple une forme de modèle de robot qui ne monte pas les marches : savoir anticiper avant même d'effectuer des essais que cette version sera caduque. J'ai bien sûr développé grandement mes connaissances en ROS, ainsi qu'en langage orienté objet **Python**. Cela me permettrait d'effectuer à nouveau un stage en robotique en cinquième année, si l'envie m'en prenait.

Durant ce stage, j'ai dû construire de A à Z mon environnement de travail ROS, modéliser seul les différentes versions du robot, ainsi qu'effectuer moi-même les simulations, conclusions et navigations pour ces dernières. J'ai réalisé la documentation de mes résultats pour un suivi interne de mon stage.

L'ambiance de travail était aussi vraiment très agréable ce qui me donnait envie de me rendre au travail tous les jours. Nous avions un groupe de stagiaire avec une très bonne entente, on passait nos pauses repas ensemble autour du babyfoot. Toute cette bonne ambiance entretient cette envie quotidienne de se rendre au travail. Ce qui est une excellente pour garder un environnement de travail sain et efficace.

Ce stage m'a aussi permis de revoir ma vision sur ce qu'était la SNCF, qui est une entreprise française innovante, elle ne se contente pas juste de gérer des trains, des gares etc. Elle participe à l'innovation française, que ce soit dans le cadre de mon stage par des robots autonomes mais aussi par des trains hybrides, à hydrogènes, des nouvelles formes de mobilités ou encore le TGV-M.

C'était pour moi une expérience très positive et, même si mon projet professionnel est de travailler dans le domaine du **big data** (data, IA, cloud, cybersécurité,...), je ne serai pas fermé à tenter une nouvelle expérience en **robotique** que ce soit au sein de la SNCF ou ailleurs.

BIBLIOGRAPHIE / WEBOGRAPHIE

Site internet de la SNCF : <https://www.sncf.com/fr>

Documentation ROS : <https://ros.org>

Documentation Gazebo : <https://gazebosim.org/home>

Documentation SDF : <http://sdformat.org>

Visuel des codes avec Carbon : <https://carbon.now.sh/>

Explication de la méthode SCRUM : <https://www.scrum.org/resources/what-is-scrum/>

Document interne de mon suivi de stage : *Détermination du meilleur concept, de ses données paramétriques et du meilleur type de navigation pour REVOLVE* - **©JulienFresnel**

Document interne : *Organigramme DTIPG* - **©SNCF**

SIGLES ET ABRÉVIATIONS

SNCF : Société nationale des chemins de fer français

TGV : Train à grande vitesse

TER : Train Express Régionaux

TFMM : Transport Ferroviaire et Multimodal de Marchandises

DTIPG : Direction Technologies, Innovations et Projets Groupe

ROS : Robot Operating System

URDF : Universal Robot Description Format

SDF : Simulation Description Format

IDE : Integrated Development Environment

FSM : Finite-State Machine

LiDAR : Light detection and ranging

GLOSSAIRE

Digitalisation : Processus de transformation des services (financiers, commerciaux) d'une entreprise, par un recours accru aux technologies de l'information.

Odométrie : technique permettant d'estimer la position d'un véhicule en mouvement.

RÉSUMÉ

Dans le cadre de mon parcours d'ingénieur, j'ai dû réaliser, pour valider ma quatrième année, un stage d'au moins 8 semaines. J'ai effectué ce stage au sein de l'équipe **Robots&Humains** de la SNCF, du 13 juin au 25 août 2022.

Durant mes **trois mois de stage**, j'ai travaillé sur un **projet de recherche** nommé **REVOLVE**, qui vise à concevoir un robot autonome parcourant les tunnels du RER E, pour aider les agents à déterminer les types de problèmes signalés. Pour cela, j'ai dû utiliser des outils comme **ROS** et faire des simulations grâce au logiciel **Gazebo**. J'ai ensuite déterminé le meilleur modèle possible et j'ai trouvé des idées de **navigation autonome**, soit par l'automatisation soit par la robotisation.

ABSTRACT

To complete my engineering course, I had to do an internship, to conclude my fourth year, for at least 8 weeks. I did it into the **Robot&Humains** team of the SNCF, from June 13th to August 25th.

During these **three months**, I worked on a **research project** named **REVOLVE**, which aims to design an autonomous robot which goes through the RER E tunnels, to help determining the types of reported issues. To realize this, I used some tools such as **ROS** or **Gazebo**, to create models and try these in simulations. I choose the best model for our case and found some ideas to code the **navigation**, either by automation or robotization.