Rapport de Stage
Cycle Ingénieur • 2ème année
2021/2022

Elève ingénieur

Prénom : Joseph

Nom : LOUVILLE

Majeure d'enseignement :

❒ SI ❒ OCRES
☒ SE ❒ FIN
❒ ENE ❒ SAN

Entreprise d'accueil

Nom : ECE Paris

Adresse : 37 Quai de Grenelle 75015 Paris

Adresse du lieu de stage si différent :

Confidentiel : ❒ oui ☒ non

Rapport à remettre au tuteur de stage à l'issue de la correction : ☒ oui ❒ non

Signature du Tuteur de Stage (**obligatoire**) :

Description de la mission

Simulation sous ROS des véhicules équipés avec des capteurs/caméras/radars ou lidars

Détection/évitement des obstacles

Planification de trajectoire

Interface avec simulateur V2X

Je tiens à remercier tout particulièrement le professeur Jae Yun JUN KIM et la professeure Naila BOUCHEMAL pour m'avoir guidé tout au long du stage et m'avoir offert cette opportunité de faire mon entrée dans le monde de la recherche. Ils ont su me montrer les méthodes et les techniques qui m'ont permis d'arriver à la présentation de mon projet de recherche.

Je remercie aussi la professeure Assia SOUKANE pour m'avoir permis d'intégrer le pôle recherche de l'ECE qu'elle dirige.

Je tiens à remercier les différents stagiaires qui étaient présents avec moi lors de ma période de stage.

Présentation de la mission :

Le stage s'est déroulé au sein de l'ECE qui est présentée ainsi par son rapport d'activité :

« Ecole d'ingénieurs du groupe OMNES Education (anciennement INSEEC U.), opérateur français privé d'enseignement supérieur, comprenant 13 institutions académiques rassemblant environ 35.000 étudiants, sur 11 campus dont 5 en France : Paris, Lyon, Bordeaux, Chambéry, Beaune et 5 internationaux : Londres, San Francisco, Genève, Monaco, Munich et Barcelone.

Dans le contexte des grands défis auxquels le monde fait face, tels que l'ONU les a résumés avec les 17 objectifs de développement durable, la mission que s'est donnée l'ECE (Ecole Centrale d'Electronique) est de « former les ingénieurs et les experts des technologies du 21e siècle à même de relever les défis de la double révolution du digital et du développement durable, sachant combiner savoirs académiques, savoir-faire de mise en œuvre, et savoir-être relationnel, au service de leur contribution au progrès économique et social qu'ils exercent au sein d'entreprises de toutes tailles et d'acteurs publics ». Elle répond aux besoins croissants en profils experts et managers dans les domaines de l'ingénierie numérique, combinaison de l'ingénierie système ; de l'informatique et des réseaux ; de la science des données et l'intelligence artificielle. Ses diplômés exercent dans tous les secteurs d'activité, la stratégie de l'école étant fortement guidée par les besoins des entreprises et acteurs publics employeurs.

Pour l'année universitaire 2021-2022, l'ECE compte près de 3.400 étudiants répartis sur ses deux campus de Paris et de Lyon.

L'Ecole a établi au fil des années une cohérence entre les profils et attentes des étudiants recrutés, la formation dispensée, ses activités de recherche et son réseau de partenaires (laboratoires de recherche, entreprises, acteurs publics, universités internationales, etc.). Les différentes enquêtes sur le placement des diplômés montrent l'intérêt porté par les entreprises pour les profils ECE et l'adéquation de la formation et des compétences acquises avec les besoins des entreprises.

L'ECE a toujours affirmé sa volonté de répondre aux besoins croissants en ingénieurs dans tous les secteurs d'activité. Pour cela, l'ECE s'appuie sur ses liens forts avec le monde économique et notamment avec les entreprises et acteurs publics ainsi que le monde de la recherche.

L'ECE se positionne comme un acteur régional et national au service de la compétitivité de la France. L'ECE est en veille permanente sur l'orientation et les évolutions des secteurs liés à son cœur de métier.

Son ancrage avec les milieux socio-économiques est fort, elle est membre actif de plusieurs réseaux nationaux et locaux d'entreprises. La forte implication des entreprises dans les enseignements, les visites de stage, le suivi des apprentis, les enquêtes et analyses pour évaluer les besoins de compétences dans des domaines cibles participent à l'identification des nouveaux besoins soit en termes d'orientations technologiques, soit en termes de volume. »

La recherche à l'intérieur de l'ECE est coordonnée et soutenue par OMNES Research Center, qui est le centre de recherche qui accueille l'ensemble des enseignants-chercheurs du groupe OMNES Education. Ce centre de recherche anime différents axes fédérateurs et transversaux aux écoles du groupe et propose des services et activités mutualisés : pôle de montage de projets européens, incubateur doctoral, Disruptive Learning Pathways Lab, collection études de cas, valorisation et communication, accords académiques, soutien aux classements et aux accréditations (partie productions académiques).

Le centre de recherche de l'ECE a été fondé en 2004, centré essentiellement sur les réseaux temps réel. Aujourd'hui, le centre compte une quinzaine d'enseignants-chercheurs permanents, une dizaine de doctorants, ainsi que de nombreux stagiaires et élèves de l'ECE. En complément, l'Ecole fait appel à des spécialistes extérieurs. En moyenne depuis 2019 cela concerne 49 enseignants-chercheurs par an dont 8 sont titulaires d'une HDR.

La mission qui m'a été assignée lors du stage consistait, dans le contexte du développement des véhicules autonomes pour des convois, dans le projet de recherche de la professeure Naila BOUCHEMAL et du professeur Jae Yun JUN KIM, de créer une plateforme robotique-communicante, c'est-à-dire de lier plusieurs simulateurs afin d'avoir un visuel complet entre l'environnement et les communications inter-véhiculaires, mais aussi la possibilité de représenter les communications avec l'environnement.

Ce projet était dans la considération d'une continuité de projet ayant déjà été fait au sein de l'école pour le challenge UTAC CERAM, qui est organisé sous l'égide de la Société des Ingénieurs de l'Automobile (SIA), dans le but de mettre en avant les nouvelles technologies de l'automobiles et les fonctions liées à l'autonomie et la sécurité.

Donc le stage proposait de travailler dans la recherche tout en amenant un aspect technique dans la compréhension de la technologie utilisée dans ce domaine et de comment amener des simulateurs à communiquer au sein de l'ordinateur, dans le but de permettre aux chercheurs qui reprendraient le projet à postériori d'avoir une plateforme pratique et adaptable pour les communications des véhicules autonomes, particulièrement dans le contexte des convois.

Possiblement, à la fin de mon stage, on m'a proposé de déposer un article sur le sujet si j'avais assez fait sur le sujet pour rendre compte d'une avancée scientifique et technique dans le domaine. Il s'agissait de mettre en avant ce travail de recherche pour mon avenir dans le monde de la recherche.

Cahier des charges et planning :

Pour parvenir à l'objectif qui m'avait été donné de mettre en place une plateforme robotique communicante, c'est-à-dire de permettre à un simulateur de robotique, où, dans le cas de mon stage, il s'agissait de ROS (Robotic Operating System), et un simulateur de réseaux de communication permettant de visualiser les transmissions entre équipements utilisateurs (« User Equipments » en anglais). Il a été décidé que pour simulateur de réseau utilisé pour le stage serait OMNeT++, car il s'agit d'un projet Open Source avec une grosse communauté derrière pour l'utilisation de ses projets.

On a mis à ma disposition un PC puissant pour la simulation avec lequel je devais développer la liaison entre les différents simulateurs, avec comme base les projets PFE (Projet Fin d'Étude) qui avaient été faits par deux équipes différentes sur les deux années précédentes, qui étaient des projets de développement de communication basés uniquement sur ROS. Tout l'intérêt donc sur le premier mois de stage devait donc consister à comprendre les différents simulateurs et comment avait été développé les différents sujets.

D'un autre côté, il fallait que je fasse un état de l'art des projets existants permettant des liaisons entre simulateurs, afin de voir si je pouvais m'en inspirer pour faciliter le développement de mon projet.

Afin d'avoir un suivi régulier de mon travail, mes tuteurs m'ont proposé de faire des réunions hebdomadaires, en fin de semaine, pour décrire ce que j'aurais fait au cours de ma semaine et pouvoir diriger mes recherches sur le sujet et voir mon avancement. Cela permettait d'avoir un moment arrangé uniquement pour parler de mon stage et permettre d'avoir de longues réunions, d'environ une heure. Je pouvais aussi dès que mes tuteurs étaient disponibles et si j'en sentais le besoin, je pouvais aller les voir directement si je bloquais sur une question ou si je ne savais plus vers quoi je devais me diriger pour la suite du projet.

Toutes les deux semaines, on m'a aussi demandé en plus des réunions hebdomadaires, de produire des rapports sur le contenu que j'avais obtenu au fil du temps. Cela m'a permis de garder une trace de tout ce que je faisais et des différentes recherches que je faisais, mais aussi de commencer à mettre un mode d'emploi des différents projets que je commençais à apprendre à utiliser.

Le suivi qui en résulterait permettrait de rendre plus accessibles les différentes fonctions pour des projets futurs, mais aussi clarifier mes idées lors des réunions et commencer à préparer le rapport de stage.

Il m'a été conseillé de commencer ma journée en faisant toutes mes activités de recherche et d'état de l'art le matin en pratiquant des expérimentations et de la simulation l'après-midi, pour que je puisse concentrer mon énergie sur la recherche le matin et ainsi être plus concentré pour la lecture, et faire une activité qui nécessite peut-être moins d'effort mental pour se concentrer en pratiquant les simulations.

Sur la fin du stage, il m'a été possible de passer en télétravail pour avancer sur mon rapport de recherche.

Description du travail :

La première réunion que j'ai eue avec mes tuteurs a donné lieu à une description en général des différents aspects du stage, en commençant par décrire l'organisation du projet de recherche, soulignant qu'à la fin du stage il m'était demandé de produire un poster de très bonne qualité qu'il me serait demandé de présenter lors d'un forum de recherche des stagiaires qui était organisé fin août, où tous les stagiaires du pôle recherche devaient présenter leur recherche faites de manière similaire. Il m'a été demandé en plus de cela de produire un mode d'emploi de mon projet et des différents dispositifs qui m'ont été utiles tout au long du stage et de créer une page GitHub compilant tout ce que j'aurais pu faire durant le stage, comprenant le code, les rapports, le poster et la bibliographie.

On m'a demandé d'être très rigoureux durant tout le stage en notant le plus de détails possibles, comme par exemple toutes les versions des logiciels et systèmes d'exploitation dont j'ai pu me servir.

Je devais comprendre la conception de l'architecture de la partie autonome de la tête d'un convoi. Elle se divise en plusieurs parties :

- Perception : C'est toute la partie capteurs qui vont pouvoir détecter l'environnement, les objets et autres véhicules que peut croiser le véhicule de tête.
- Localisation : Il s'agit de comment le véhicule se place dans le trafic.
- Planification : Le véhicule va déterminer à partir de toutes les informations dont il dispose, la trajectoire la plus rapide tout en étant la moins risquée pour atteindre sa destination.
- Contrôle : Le véhicule va modifier sa vitesse et sa trajectoire pour appliquer les commandes de planification.

A partir de ces informations je devais comprendre comment les communications étaient faites entre véhicules, avec pour base, pour les anciens projets, les communications LTE ASN1, qui sont un standard pour les communications cellulaires.

J'avais comme direction de tester des projets sur OMNeT++ pour comparer les communications LTE et comparer avec les communications 5G, avec la bibliothèque open source SIMU5G qui permet de simuler un réseau complet 5G. L'idée derrière cela et peut-être dans la contribution que je pourrais apporter étaient d'investir les communications 5G dans le convoi de véhicules autonomes dans un projet ROS, avec l'objectif de lier les deux plateformes.

Au début l'objectif était donc de comprendre les projets PFE fait sur ROS les 2 dernières années. Pour cela déjà je devais comprendre comment était organisé ROS en soit et comment je pouvais faire exécuter les projets utilisant cette technologie.

ROS est donc un ensemble d'outils informatiques sous forme de logiciels libres (le système est sous licence BSD) open source, permettant de développer des logiciels pour la robotique. ROS va générer un exécutable qui va ensuite se lancer sur l'application graphique Gazebo qui va générer l'environnement à partir de fichiers 3D puis permet d'exécuter le scénario appelé pour la simulation.

Finalement lors de cette réunion, nous avons discuté de la méthode avec laquelle j'allais analyser tous les différents articles de recherche pour mon état de l'Art. En effet, chaque article doit être lu de la même manière, en commençant par la problématique de recherche puis il faut comprendre la méthodologie qui est utilisée pour comprendre ce qui est recherché dans l'application de l'étude.

Puis il faut trouver la contribution qu'apporte l'article, ce qui permet de comprendre les résultats, et on en ressort à partir de ça l'impact que peut avoir cette étude. Puis enfin il faut en ressortir un avis critique afin de montrer les points intéressants de la recherche et vraiment trouver quel intérêt on peut vraiment en ressortir.

A partir de là, j'ai commencé à étudier les projets PFE afin de comprendre comment la technologie ROS fonctionne et comment les projets sont constitués.

J'ai eu de nombreux problèmes pour les faire fonctionner, car en essayant de juste appliquer la méthode qui était donnée dans les rapports de PFE, il s'est passé qu'il manquait une grande partie des paquets requis pour faire fonctionner ROS, ainsi que des paquets supplémentaires pour ces projets.

J'ai donc dû apprendre à installer tous les processus de ROS afin de pouvoir manipuler ces projets et comprendre leur fonctionnement.

Mais il s'est avéré que même en ayant fait cette étape, il manquait des paramètres dans les fichiers d'exécutions, qui avaient été fait par l'équipe à l'origine du projet initial.

Pour résoudre ce problème, je suis, d'un côté, entré en contact avec un membre de cette équipe, Alexandre Ségarat, que j'avais pu rencontrer auparavant, ce qui m'a aidé pour le contacter pour poser des questions sur ce sujet. D'un autre côté, je suis allé regarder des tutoriels et documentations sur ROS et la création de paquets customisés, puisque ces projets se basent essentiellement dessus, comme j'ai pu le constater durant cette première période de stage.

J'ai pu obtenir de ça, après avoir trouvé comment résoudre ces problèmes, un mode d'emploi pour utiliser ces projets et j'ai pu détailler en partie le fonctionnement de ROS que j'ai pu voir lors de la découverte de ces créations.

J'ai donc pu avoir une version identique à celle des rapports sur l'ordinateur fourni par le centre recherche, qui était donc le même pc qui avait été utilisé par les équipes précédentes, sur Ubuntu 16.04, avec la version Kinetic de ROS, publiée en 2016. Dans l'idée, cela m'a pris environ deux semaines pour arriver à ce résultat.

Il m'a ensuite été demandé de créer une machine virtuelle pouvant exécuter ces projets, ce qui aura aussi été une difficulté importante, déjà parce que je n'avais jamais eu l'occasion de voir comment on pouvait faire une telle procédure ; mais aussi parce qu'après avoir regardé comment faire à partir de la partition système, il est impossible de virtualiser la partition actuellement utilisée, donc il est conseillé de soit créer une machine virtuelle vide et de faire les installations directement sur cette machine virtuelle ou de passer par une deuxième partition installée sur le PC.

N'ayant pas encore beaucoup appris sur le partitionnage d'un PC à ce moment, j'ai préféré prendre la première solution pour ne pas perdre trop de temps et ainsi enfin pouvoir passer à la suite.

Ce que j'avais obtenu a semblé plaire à mes tuteurs et je pense que cela a apporté en soit une avancée rapide sur mon travail, étant donné la complexité des systèmes à apprendre et les installations nécessaires pour tous les fonctionnements des différents projets.

Après cela, on m'a demandé de voir comment fonctionne OMNeT++, qui est donc ce simulateur de réseaux de communication qui est central dans le choix de création de la plateforme robotique communicante. Avec ça, l'objectif était de mettre en place de quoi pouvoir faire fonctionner sur le PC le projet de réseau cellulaire Simu5G. En même temps de faire mes installations et comprendre comment fonctionne OMNeT++, qui est basé sur le langage de programmation C++, il m'a été demandé de faire

une analyse d'un article qui a pour but de présenter dans les grandes lignes la technologie LTE et ce qu'apporte la 5G dans les communications.

La cinquième génération des réseaux cellulaires, la 5G est attendue pour une transformation radicale du paysage des réseaux. De cette technologie vient se rajouter le Multi-access Edge Computing (MEC), qui vient ajouter les systèmes de clouds en bordure des réseaux de communications.

La technologie de Simu5G propose une simulation end-to-end d'application de communications complexes, des scénarios hétérogènes, et qui permet la coexistence de la 4G et 5G.

Simu5G est basé sur le simulateur simuLTE, qui permet l'évaluation de performance des réseaux LTE et LTE Advanced pour le système OMNeT++. SimuLTE est écrit en C++ et est entièrement customisable. L'évaluation des performances des réseaux 5G est d'une importance primordiale, car il y a un grand besoin de concevoir et de valider des projets de gestion des ressources pour la 5G RAN (Radio Access Network).

Par exemple, la répartition des ressources pour la connexion double est à la fois la 4G et la 5G des UE (User Equipments), ou les algorithmes de planification des gNBs. Aussi la 5G devient importante pour le développement de systèmes dépendant de latences critiques, les systèmes temps réel, on a besoin de vérifier la faisabilité et les performances de l'utilisation de services basés sur la 5G. Il existe d'autres simulateurs de 5G, 5G-LENA, 5GK-Simulator, Vienna 5G SL Simulator et WiSE, mais ils ne proposent pas de simuler des paquets d'application qui circulent dans le réseau.

Simu5G prend en compte la bibliothèque INET, qui permet de simuler des réseaux TCP/IP génériques incluant des interfaces 2-couches NR (New Radio) 5G. En particulier, Simu5G simule le plan de données du 5G RAN et du réseau central. Il permet la simulation des communications 5G à la fois dans la division de fréquence Duplexage (FDD) et duplexage par répartition dans le temps (TDD), avec des gNBs hétérogènes (macro, micro, pico, etc.), communiquant éventuellement via l'interface X2 pour prendre en charge le transfert et les interférences intercellulaires coordonnées. Une double connectivité entre un eNB (LTE station de base) et une gNB (station de base 5G NR) est également disponible.

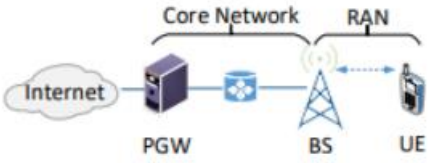
Les couches de protocole conformes au 3GPP sont fournies, tandis que la couche physique est modélisée via des modèles de canaux réalistes et personnalisables. La planification des ressources dans les sens montants et descendants est prise en charge, avec prise en charge de l'agrégation de porteuses et de plusieurs numérologies, comme spécifié par la norme 3GPP (3GPP TR 38.300, TR 38.211).

Simu5G supporte une grande variété de modèles de mobilité des UE, y compris mobilité véhiculaire. De plus, il permet d'instancier des scénarios où une application utilisateur, en cours d'exécution sur l'UE, communique avec une application MEC résidant sur un hôte MEC, pour évaluer (par exemple) la latence aller-retour d'un service de nouvelle génération, y compris le temps de calcul au MEC hôte. Plus précisément, Simu5G peut fonctionner en mode d'émulation temps réel, permettant l'interaction avec de vrais appareils.

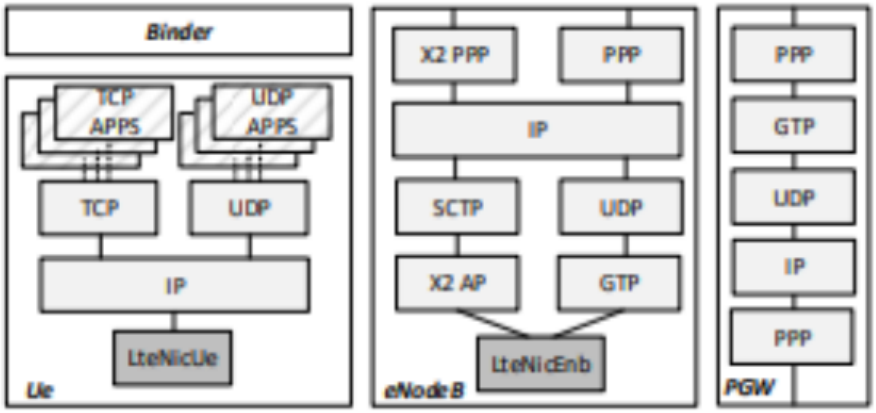
En fait, d'une part OMNeT++ permet la programmation d'événements en temps réel ; d'autre part, la librairie INET permet d'être paramétrée pour échanger des paquets IP entre les applications locales ou les interfaces réseau et le simulateur. Un réseau cellulaire se compose d'un RAN et d'un réseau central. Le RAN est composé de cellules, sous le contrôle d'une seule station de base (BS).

Les UE sont attachées à une BS et peuvent changer la BS de desserte via une procédure de transfert. Les BS communiquent entre elles via l'interface X2, une connexion logique qui fonctionne normalement sur un réseau câblé. Le réseau central est constitué de routeurs IP reliant un point d'entrée, le Packet Gateway (PGW) aux BS. La transmission dans le réseau central s'effectue à l'aide du protocole de

tunnellisation GPRS (GTP). Dans le RAN, les communications entre la BS et l'UE se produisent au niveau de la couche 2 du modèle de référence OSI. Les couches 1 et 2 sont mises en œuvre à l'aide d'une pile de quatre protocoles, à la fois sur la BS et sur l'UE. De haut en bas, nous trouvons d'abord le protocole PDCP (Packet Data Convergence Protocol), qui reçoit les datagrammes IP, effectue le chiffrement et la numérotation, et les envoie à la couche RLC (Radio Link Control).



Les unités de données de service (SDU) RLC sont stockées dans la mémoire tampon RLC et sont récupérées par la couche MAC (Media Access Control) sous-jacente lorsque cette dernière doit composer une transmission. Le MAC assemble les unités de données de protocole RLC (PDU) en blocs de transport (TB), ajoute un en-tête MAC et envoie tout via la couche physique (PHY) pour transmission. SimuLTE simule le plan de données du RAN et du réseau central LTE/LTE-A.



Les UE et les BS (appelés eNB dans LTE) sont implémentés sous forme de modules composés, qui peuvent être connectés les uns aux autres et à d'autres nœuds (par exemple, des routeurs, des applications, etc.) pour composer des réseaux. Les deux ont une carte d'interface réseau LTE (NIC), qui implémente la pile de protocoles LTE. Le module UE intègre également les protocoles IP et TCP/UDP, ainsi que des vecteurs d'applications TCP/UDP.

Le module eNodeB comprend deux interfaces PPP : le module X2PPP permet la connexion directe avec les eNB voisins utilisant le protocole Stream Control Transmission Protocol (SCTP) tel que spécifié par la norme, tandis que le module PPP est connecté au module PGW. SimuLTE simule le plan de

données du RAN et du réseau central LTE/LTE-A. Les UE et les BS (appelés eNB dans LTE) sont implémentés sous forme de modules composés, qui peuvent être connectés les uns aux autres et à d'autres nœuds (par exemple, des routeurs, des applications, etc.) pour composer des réseaux. Les deux ont une carte d'interface réseau LTE (NIC), qui implémente la pile de protocoles LTE. Le module UE intègre également les protocoles IP et TCP/UDP, ainsi que des vecteurs d'applications TCP/UDP. Le module eNodeB comprend deux interfaces PPP : le module X2PPP permet la connexion directe avec les eNB voisins utilisant le protocole Stream Control Transmission Protocol (SCTP) tel que spécifié par la norme, tandis que le module PPP est connecté au module PGW.

Je me suis posé la question des différentes versions de ROS car les versions utilisées pour les projets de l'ECE commençaient déjà à être archaïque, et des versions plus complexes et plus intéressantes commençaient à sortir récemment, dont particulièrement ROS 2, qui permet de contrôler facilement en temps réel plusieurs appareils en même temps grâce aux nouveaux standards de communications qui sont sortis pour les réseaux de communications industrielles et équipements qu'est le Data Distribution Service.

Cette nouvelle technologie de programmation de robotique est très intéressante pour les convois de véhicules autonomes, mais il a été décidé de se concentrer sur des projets déjà existants pour la continuité du stage de recherche.

Après cela on m'a demandé de faire l'état de l'Art sur les projets de recherche existants traitant de la liaison entre ROS et OMNET++, tout en recherchant sur Simu5G les propriétés intéressantes pour possiblement à terme faire une liaison entre un projet ROS et Simu5G.

Wireless Characteristics Study for Indoor Multi-Robot Communication System:

Projet basé sur le modèle de Tata Indoor Path Loss Model (T-IPLM).

Etudie le SNR selon les obstacles et les Access Points. L'environnement physique de ROS est partagé avec OMNeT par la configuration appelée omnetpp.ini. La bibliothèque INET ajoute aux nœuds ROS des fonctionnalités de carte d'interface réseau (NIC) sans fil qui mettent en œuvre les modules radio et MAC. Chaque NIC communique ensuite avec l'application ROS par le biais du module ROSForwarder de OMNeT.

En utilisant le module ROSForwarder, nous créons un service ROSS appelé packetSenderService qui convertit les messages du nœud ROS expéditeur en paquets et attache l'adresse MAC du nœud ROS destinataire. Le paquet est transmis par le RadioMedium qui prend en charge la propagation, la perte de chemin, la perte d'obstacle, etc. dans l'environnement.

Lorsque le paquet atteint la destination, le NIC du nœud ROS détermine, en fonction du support radio sous-jacent et de la sensibilité du récepteur, si le paquet a été reçu avec succès ou non. L'ensemble de la configuration de simulation entre ROS et OMNeT est synchronisé par le module ROSSynchonizer d'OMNeT. Il a également été considéré deux des modèles de canaux existants, tels que Log Normal et ITU-R d'OMNET, et inclus le modèle T-IPLM pour comparer les comportements sans fil.

Simulation Framework for Platooning based on Gazebo and SUMO:

Mise en place de simulation d'un convoi avec Sumo sur ROS et Gazebo. Les différentes méthodes possibles sont proposées dans l'article. Le modèle proposé est divisé en trois parties. Le véhicule de tête, l'ego-véhicule, est codé en Python sur ROS, pour fonctionner dans un environnement SUMO. La communication entre véhicule est faite par les topics de ROS, donc idéale.

Le véhicule est donc commandé par l'intermédiaire de SUMO. Cet article ne parle pas de la limite de vitesse de transmission et les véhicules dans le convoi ne s'adaptent pas vraiment au message que fait le véhicule de tête, mais ne font que reproduire les déplacements de celui-ci.

COPADRIVe - A Realistic Simulation Framework for Cooperative Autonomous Driving Applications:

L'article met en avant la technologie de message ETSI ITS-G5, une technologie qui pourrait amener rapidement le déploiement de V2X.

Il propose une architecture du framework de la liaison entre ROS et OMNeT++ par le biais de Gazebo et Artery. Artery permet d'intégrer Veins qui connecte SUMO et OMNeT++ et qui permet d'intégrer Vanetza ITS-G5.

La liaison repose sur le sub/pub pour intégrer OMNeT++ et Gazebo. Chaque nœud d'OMNeT++ représentes une interface de réseau d'une voiture et contient un fournisseur de données de véhicule et un robot middleware. Le fournisseur de données de véhicule fait le lien entre le robot middleware et Gazebo.

Le robot middleware se sert des informations pour remplir les champs de données des CAMs de l'ITS-G5 à travers le CaService qui va encoder les champs de données pour correspondre aux définitions ITS-G5 ASN-1.

Le robot middleware transmet aussi les coordonnées GPS du module de mobilité d'INET. OMNeT++ étant un simulateur événementiel et Gazebo un simulateur temporel, la synchronisation des deux simulateurs représentait un défi majeur. Pour y parvenir, un module de synchronisation a été implémenté dans OMNeT++, afin de réaliser cette tâche, en s'appuyant sur le sujet "/Clock" de ROS comme référence d'horloge.

Le module de synchronisation d'OMNeT++ s'abonne au sujet ROS "/Clock", publié à chaque étape de la simulation Gazebo (c'est-à-dire toutes les 1ms) et procède à la programmation d'un message OMNeT++ fait sur mesure à cet effet ("syncMsg") à une heure ROS exacte, ce qui permet au moteur du simulateur d'OMNeT++ de générer un événement lorsqu'il atteint cet horodatage et ainsi d'être en mesure de poursuivre tout autre processus de simulation qui devrait s'exécuter au même moment (par exemple, la génération de CAM par CAService).

Ensuite il m'a été demandé de voir les capacités de COPADRIVe, ce qui a semblé être sur le papier un projet pertinent dans la suite de mes objectifs, puisqu'il mettait en place les communications en parallèle avec ROS, ce qui faisait que l'on avait la transmission des messages CAMs à la fois sur ROS et sur OMNeT++. Or l'un de mes tuteurs, Madame BOUCHEMAL, m'a fait comprendre que les communications par le biais de CAM pouvaient être très important dans la suite de mon projet, car c'était la base de communications standardisées qu'elle avait déjà essayée de mettre en place avec les groupes PFE précédents.

Pour réussir à installer COPADRIVe, il était recommandé sur la page GitHub du projet d'être sur la version 18.04 d'Ubuntu, donc si je voulais la faire fonctionner sur mon PC, il fallait que je voie comment le repartitionner. J'ai regardé des tutoriels et de la documentation sur différents site internet, mais il m'a fallu prendre du temps pour tout installer car j'ai dû faire face à plusieurs bugs qui m'ont ralenti dans ma progression des installations, surtout qu'on m'avait demandé donc de partitionner plusieurs versions d'Ubuntu sur le PC, afin de tester les différents logiciels sur différentes versions.

J'ai passé plusieurs semaines à comprendre COPADRIVe, qui est un projet complexe dont je me suis rendu compte qu'il n'était pas pratique à adapter aux fichiers, déjà parce que la rare documentation technique sur le projet est écrit en portugais, mais passée la barrière du langage, les indications sur le fonctionnement du projet sont assez pauvres en informations, et il m'a fallu presque un mois pour juste comprendre quels fichiers du projet permettaient de faire la communication entre ROS et OMNeT++.

Utilisation de ROS dans la simulation des véhicules autonomes :

Les projets de véhicules autonomes proposent des modèles faciles à appliquer en C++. ROS propose de communiquer de bases par des topics en sub/pub transmissions. Mais il est possible de customiser la plupart des éléments, de la communication aux éléments de l'environnement comme l'appareil en lui-même.

Le projet Copadrive propose des simulations sur circuits ou des véhicules autonomes basés sur les Toyota Prius comme modèle physique et la communication entre les véhicules est simulée sur OMNeT++. Le projet ECE a pour but de simuler les communications 4G et 5G sur ROS en créant des messages inspirés des messages DENM et SPAT.

Plusieurs modèles d'architecture qui permettent la communication entre ROS et OMNeT++ par les biais des topics que proposent ROS par les systèmes de Pub/Sub. Certains modèles se servent de SUMO, qui est un simulateur de trafic dans un espace routier défini.

Par le biais d'OMNeT, on peut donc reproduire les communications en direct dans un réseau routier. CopaDrive est un framework qui permet de lier ROS et OMNeT en se basant sur Artery, un modèle avancé pour les réseaux de communications pour les véhicules fonctionnant sur OMNeT++ et SUMO.

Il a donc été décidé que je mette en place plusieurs versions sur le PC afin de tester chaque projet comme suivant :

	Ubuntu	ROS	Gazebo	Veins	Simu5G	INET	OMNeT++
1	18.04	Melodic	9	Artery	X	3.7.1	5.5.1 min
2	16.04	Kinetic	7	X	X	X	X
3	20.04	X	X	5.1	1.2.0	4.3.2	6.0(pre10 et pre11)
4	18.04	Melodic	9	artery	1.1.0	4.2.2	5.6.2

Le 1 correspond au projet CopaDrive comme il est téléchargeable sur Github. Le 2 est le projet ECE qui a été fait en 2020. Le 3 correspond est le modèle de simulation de Simu5G sur OMNeT++. Le 4 correspond àn la version Objectif, contenant l'ensemble de ces concepts.

A partir du moment où j'ai eu ces logiciels d'installés, l'objectif a été de comprendre plus en profondeur COPADRIVe, au niveau de chaque liaison quel fichier décrivait telle ou telle action, dans le but de pouvoir reproduire une telle liaison, étant donné que les communications basées sur le standard ITS-G5 étaient indispensables pour l'avancement du projet.

Le but ayant légèrement dévié de sa trajectoire puisque COPADRIVe correspond à la plateforme que l'on souhaitait obtenir pour les communications entre simulateurs, l'intérêt était plus de pouvoir reproduire, étudier les communications pour comprendre si elles étaient plus à même d'être interpréter dans une vision réaliste pour les échanges inter-véhiculaires. Aussi il était intéressant de comprendre quelles informations étaient transmises par les messages CAMs, s'il n'y avait que ces messages qui étaient transmis, et si tel était le cas, comment implanter d'autres messages, comme les DENMs, qui ont plus un intérêt pour les échanges et la détection de l'environnement.

J'ai donc passé plusieurs semaines à tester, interpréter et modifier les fichiers qui composent COPADRIVe, en passant par les fichiers ROS, jusqu'à ceux d'OMNeT++.

COPADRIVe Architecture :

La pile protocolaire de CopaDrive est basée sur ITS-G5 par l'intermédiaire de Vanetza, qui reproduit en simulation Vanet (Vehicular Ad-Hoc Network) qui permet de faire la communication entre les véhicules et les équipements fixes à portée (V2X).

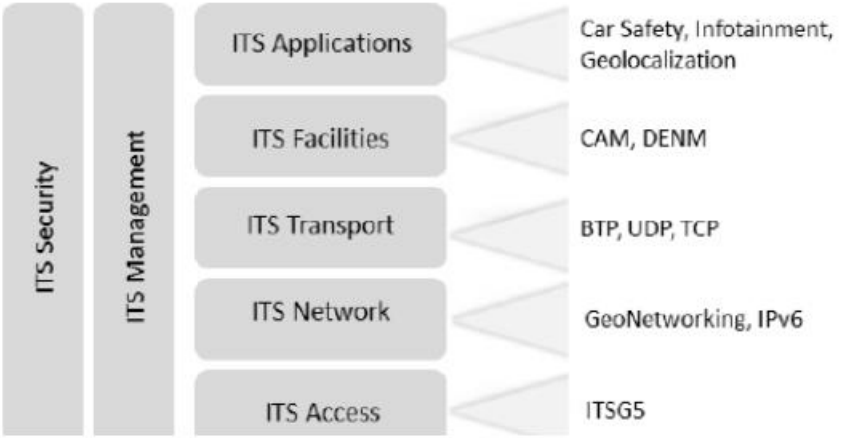


Figure 1: Protocol architecture d'ITS-G5

Dans Artery, on peut trouver des fichiers correspondant à l'utilisation de messages CAM, DENM et SPAT. Or lors du test d'utilisation de COPADRIVe, seulement les messages sont affichés et utilisés sur OMNeT++.

En ajoutant un obstacle dans la trajectoire des véhicules sur COPADRIVe, on observe que la fonction de détection de la ligne pour le déplacement prend aussi en compte les objets et permet de les contourner. Il est possible de rajouter des objets dans gazebo en utilisant la fonction insertion.

Fonctionnement CAM :

Les messages de sensibilisation coopératifs (CAM) sont distribués dans le réseau ITS-G5 (802.11p) et fournissent des informations sur la présence, la position et l'état de base des stations ITS communicantes aux stations ITS voisines situées à une distance d'un seul saut. Toutes les stations ITS doivent pouvoir générer, envoyer et recevoir des CAM, pour autant qu'elles participent à des réseaux V2X.

En recevant des CAM, la station STI est informée des autres stations situées dans son voisinage ainsi que de leur position, de leurs mouvements, de leurs attributs de base et des informations de base sur les capteurs.

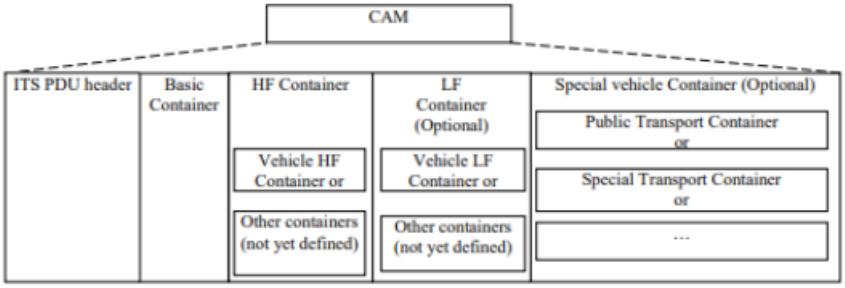


Figure 2: Structure générale d'un CAM.

Du côté du récepteur, des efforts raisonnables peuvent être faits pour évaluer la pertinence des messages et des informations. Cela permet aux stations ITS d'obtenir des informations sur leur situation et d'agir en conséquence.

La gestion des CAM est indépendante des applications. C'est pourquoi il n'y a pas d'interface avec les applications.

- Intervalle de temps maximum entre les générations de CAM : 1 s
- Intervalle de temps minimum entre les générations de CAM : 0,1 s

Ces règles sont vérifiées au plus tard toutes les 100 ms ; générer un CAM lorsque la différence absolue entre le cap actuel (vers le Nord) et le dernier cap CAM > 4° ; générer un CAM lorsque la distance entre la position actuelle et la dernière position CAM > 5 m ; générer une CAM lorsque la différence absolue entre la vitesse actuelle et la vitesse du dernier CAM > 1 m/s ; les règles de génération sont vérifiées toutes les 100 ms.

Fonctionnement DENM :

Un DENM contient des informations relatives à un événement qui a un impact potentiel sur la sécurité routière ou les conditions de circulation.

Un événement est caractérisé par un type d'événement, une position d'événement, un temps de détection et une durée. Ces attributs peuvent changer dans l'espace et dans le temps. Dans certaines situations, le STI-S d'origine transmet un DENM d'un événement causé par le ITS-S lui-même, tel qu'un événement de feu de freinage électronique.

L'ITS-S d'origine gère la transmission et la fin du DENM pour cet événement. Cependant, dans d'autres situations, les DENM liés au même événement peuvent être transmis par plus d'un ITS-S d'origine. ITS-S d'origine.

En outre, dans le cas où l'ITS-S d'origine est mobile (par exemple, un ITS-S de véhicule ou un ITS-S personnel), un événement peut persister même après que l'ITS-S d'origine se soit déplacé vers une position éloignée de la position de l'événement. Par exemple, plusieurs véhicules ITS-S peuvent détecter du verglas sur la chaussée et transmettre des DENM.

Ces DENMs sont relayées par d'autres ITS-S même d'autres ITS-S, même après que les ITS-S détecteurs de véhicules ont quitté l'emplacement du verglas. Par conséquent, la transmission des DENM est indépendante de l'ITS-S d'origine dans cet exemple.

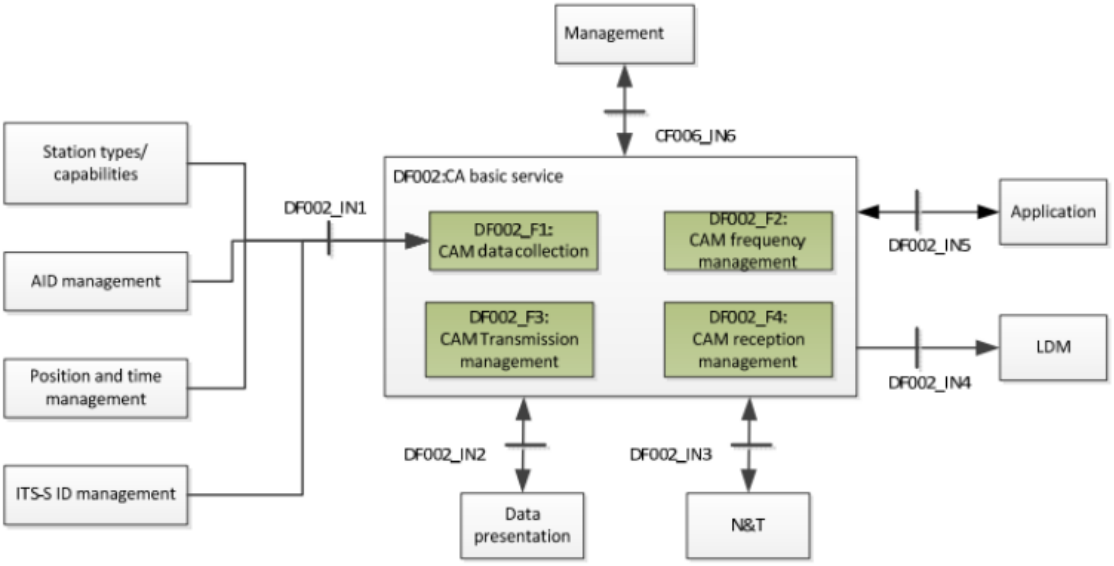


Figure 3: DEN fonctionnement

Le protocole DENM est conçu pour gérer ces situations. Les types de DENM suivants sont définis :

- Nouveau DENM : Un DENM généré par le service de base DEN lorsqu'un événement est détecté par un ITS-S d'origine pour la première fois. Chaque nouveau DENM se voit attribuer un nouvel identifiant, dénommé actionID. Un nouveau DENM fournit des attributs d'événement, tels que la position de l'événement, le type d'événement, l'heure de détection de l'événement, et d'autres attributs.

- DENM de mise à jour : un DENM généré par le service de base DEN qui comprend des informations de mise à jour d'un événement. Un DENM de mise à jour est transmis par le même ITS-S d'origine qui a généré le nouveau DENM pour le même événement.

- DENM d'annulation : Un DENM qui informe de la fin d'un événement. Un DENM d'annulation est transmis par le même ITS-S d'origine qui a généré le nouveau DENM pour le même événement.

- DENM de négation : DENM qui informe de la fin d'un événement pour lequel le nouveau DENM a été reçu par l'ITS-S d'origine en provenance d'un autre ITS-S. Un DENM de négation peut être utilisé pour annoncer la fin d'un événement si l'ITS-S d'origine a la capacité de détecter la fin d'un événement qui a été précédemment annoncé par d'autres ITS-S. Par exemple, l'ITS-S d'origine d'un nouveau DENM indiquant que du verglas a quitté la position de l'événement, quelque temps plus tard, un autre ITS-S recevant ce nouveau DENM atteint la position de verglas indiquée et détecte que le verglas a disparu. Ce dernier ITS-S peut dans ce cas générer

un DENM de négation pour cet événement. La transmission d'un DENM de négation peut dépendre des exigences de l'application et du déploiement.

Le service de base DEN de l'ITS-S d'origine doit être capable de construire les types de DENM ci-dessus. L'application ITS-S de l'ITS-S d'origine envoie une demande d'application au service de base DEN afin de déclencher la génération de DENMs. Le type de DENM à générer dépend du type de la demande d'application.

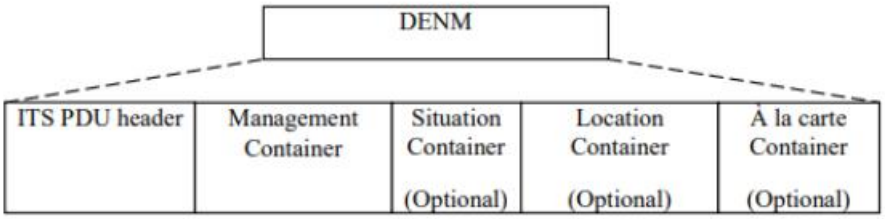


Figure 4: Structure générale d'un DENM.

Dans Artery on retrouve les fichiers DEN où sont générés les CauseCode qui vont provoquer la transmission de message DENM. Ces fichiers font appel aux définitions faites sur Vanetza, qui reproduit l'architecture de Vanet.

Etant donné que le stage commencait à arriver sur la fin à partir de ce moment, on a commencé à revoir les discussions sur les objectifs de la fin de stage, car je commençais à manquer de temps. En effet, je m'étais rendu compte au fur et à mesure que COPADRIVe était difficilement modifiable pour d'autres fichiers, car le projet avait été visiblement adapté pour uniquement le projet ROS, et que chaque partie du projet communiquait à des points précis avec le projet ROS.

Donc il a été décidé d'abord de voir si je ne pouvais pas rajouter dans COPADRIVe les messages DENM pour donner un point de valorisation de projet de recherche, comme ça il y aurait la possibilité de tester un autre point de communication dans COPADRIVe, et peut-être pouvoir donner des résultats de performances des communications des véhicules, et permettre au convoi de véhicule autonome d'adapter rapidement la trajectoire des véhicules par rapport au contexte dans lesquels ils circulaient.

Or pour voir comment mettre en place ce genre de connexion dans un programme ROS ou OMNeT++, je n'ai pas trouvé de documentations qui puissent expliquer comment mettre en place de telles communications. Il existe des projets sur GitHub suggérant un début d'ébauche pour les messages DENM ou d'autres, mais je n'ai pas trouvé à ce moment de vraie direction pour créer le même type de communication qu'avec les CAMs, comme cela a été fait dans COPADRIVe.

Sur la fin de mes recherches, j'ai découvert qu'il existait un projet qui avait été publié récemment sur GitHub avec un article de recherche qui était lié, du nom de « AuNa : Modularly Integrated Simulation Framework for Cooperative Autonomous Navigation ». Ce projet, qui a été en grande partie inspiré par COPADRIVe, propose une version beaucoup plus intéressante et beaucoup plus adaptée pour les convois de véhicules autonomes.

Il apporte déjà ROS 2 qui représente un moyen de programmer de manière parallèle des modules de robotique et donc promet le contrôle de plusieurs appareils, grâce au standard DDS (Data Distribution Services), et donc permet des communications plus simples à mettre en place.

Ce projet promet donc de mettre en avant les nouvelles technologies de communications entre véhicules de manières à rendre plus simple les simulations de convoi et permet aussi d'analyser plus facilement les données qu'apporte les simulateurs grâce à Matlab et Simulink.

Hélas j'ai découvert ce projet un peu tard à ce stade du stage, donc avec mes tuteurs nous avons pris la décision de rester sur les projets que nous connaissions, tout en faisant en sorte de garder ce projet en vu pour peut-être de futurs sujets de recherche qui suivront celui-ci.

Je me suis rendu compte que pour commencer à préparer mon poster et ma présentation pour le forum de la recherche organisé à la fin du stage pour tous les stagiaires, j'allais manquer de méthodologie, d'expérimentations et de résultats à présenter, donc on a pris la décision, avec mes tuteurs, qu'il serait mieux pour avoir cette présentation que je passe directement à une étude comparative entre COPADRIVe et le projet PFE, comparer les performances des deux projets pour voir l'impact de l'apport d'OMNeT++ pour un projet d'une plateforme robotique communicante.

Pour cela, j'ai dû récupérer des données de déplacement des trajectoires des véhicules, des différentes commandes de trajectoire données par les messages CAMs et il fallait générer des valeurs d'erreurs de trajectoires, afin de voir quel projet donne de meilleures valeurs de trajectoires et comment elles sont bien suivies par les commandes de véhicules sur ROS.

Mais cette étape était aussi difficile car pour trouver des méthodes adaptées pour récupérer les données pour l'analyse, il a fallu que je travaille encore plus sur la compréhension de ROS et aussi sur les fichiers de COPADRIVe car les données récupérables étaient dispersées et donc difficiles d'accès.

Après cela il a fallu que je compile et compare les deux projets avec les données. Le but était d'appliquer la formule de l'erreur quadratique normalisé :

$$\boldsymbol{NMSE(normal\ mean\ square\ error) = 10log_{10}(\frac{\sum_{t=1}^{T}|e(t)|^2}{\sum_{t=1}^{T}|r(t)|^2})}$$

A partir de cela, j'ai pu créer mon poster qui décrivait rapidement le contexte du pourquoi de ce projet de recherche sur la conception d'une plateforme robotique communicante avait été mis en place. Avec ça j'ai présenté tout le travail que j'ai produit dans tout le stage, afin d'avoir une vision assez générale de ce qui avait pu amener à la problématique qui s'était posée lors de la création du poster.

De cette problématique, j'ai pu dégager des objectifs et une méthodologie que j'explique dans ce poster pour obtenir mes résultats. Le constat est qu'on obtient une valeur de NMSE plus faible pour COPADRIVe que pour le projet PFE, ce qui signifie que le projet PFE représente une valeur de précision plus intéressante, ce qui permet de dire que ce projet est plus précis.

J'ai donc eu à présenter ce poster devant les docteurs qui étaient présents lors de l'événement organisé par le centre de recherche. J'ai été noté sur cette présentation, et je pense que ça a plu à ce jury, étant donné qu'ils m'ont donné 17 sur 20 pour cette présentation.

Conclusion :

A la fin du stage, j'ai pu obtenir une analyse assez complète des projets que j'ai pu étudier tout long au long du stage ainsi qu'un état de l'Art assez complet qui pourra être utile pour les projets futurs.

J'ai réalisé un poster de recherche, et une recherche a donc bien pu être réalisée et aboutie sur ce sujet. Les projets PFE qui avaient déjà été accomplis au sein de l'ECE m'ont permis d'atteindre un nouveau degré de compétences sur les domaines de la robotique et des communications.

J'ai pu mettre en avant des technologies jamais utilisées pour la recherche sur ce domaine au sein de l'ECE pour ces projets.

J'ai aussi produit un mode d'emploi des différents projets qui m'ont été demandé d'analyser et d'utiliser. J'ai partitionné le PC de simulation des projets liés donc au projets UTAC CERAM afin de permettre de simuler sur différentes versions d'Ubuntu.

Ce que j'aurais produit comme travail d'analyse et l'acquisition de compétences qui en a découlé tout au long du stage vont permettre de produire un projet pour le concours de l'UTAC CERAM pour l'innovation sur l'automobile. Aussi normalement ce projet sera réalisé dans les conditions d'un PFE afin de faire participer les élèves de l'école.

Un des intérêts de ce sujet aura été aussi de mettre en avant les capacités du centre de recherche de l'ECE Paris à créer de l'innovation. Et il a aussi permis de faire avancer des projets au sein de l'école qui avaient jusque-là peu avancé. Il s'inscrit dans la continuité d'un projet de recherche plus grand sur le développement des convois de véhicules autonomes.

ANNEXE :

Conception d'une plateforme robotique communicante pour un convoi autonome

Joseph LOUVILLE, Naila BOUCHEMAL, Jae Yun JUN KIM
ECE Paris

CONTEXTE

- De nombreux simulateurs de robotique et de communications ont été développés ces dernières années pour le développement des véhicules autonomes : ROS (Robot Operating System), OMNeT++, SUMO.
- Des plateformes polyvalentes, robotiques communicantes, créées avec ces simulateurs, permettent d'avoir une visualisation des véhicules et des communications (5G NR (New Radio), ITS-G5 (Intelligent Transport System - G5)).
- Les convois comportent de nombreux avantages, en réduisant la consommation ou facilitent la distribution du trafic routier, pouvant faciliter des livraisons en milieu urbain, par exemple en organisant la répartition des véhicules dans l'ensemble du trafic.

ÉTAT DE L'ART

Robot Operating System (ROS) The Complete Reference [1]

Ce livre vient à présenter le fonctionnement de ROS et présenter des projets importants qui l'utilisent dans la modélisation.

OMNeT++

Un simulateur Open-Source de réseaux de communication, mais permet aussi de simuler des systèmes informatiques, des réseaux de file d'attente et des architectures hardware.

Simu5G-An OMNeT++ Library for End-to-End Performance Evaluation of 5G Networks [2]

Il s'agit d'un article visant à introduire la bibliothèque SIMU5G, qui constitue des modèles de simulation pour OMNeT++. Il permet aussi de lier SUMO avec la 5G New Radio.

COPADRIVe - A Realistic Simulation Framework for Cooperative Autonomous Driving Applications [4]

Ce projet met en place une plateforme communicante entre ROS et OMNeT++, par le biais de la bibliothèque Artery.

PROBLÉMATIQUE

Comment concevoir une plateforme robotique communicante (PRC) pour un convoi autonome?

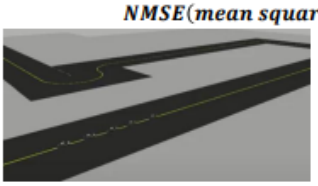
OBJECTIF

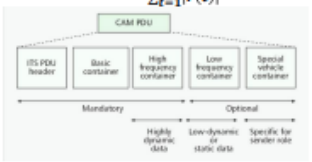
- Déterminer quelles architectures sont appropriées pour le développement de la PRC pour un convoi autonome.
- Implémenter un convoi autonome à partir de ces plateformes
- Mesurer les performances des véhicules si on apporte ce concept de PRC.
- Comparer les taux d'erreurs entre la plateforme de référence sans simulateur communicant et une PRC.

MÉTHODOLOGIE

Configuration :
- ROS Melodic
- Ubuntu 18.04 64bits
- GPU Nvidia GeForce RTX 3070
- CPU Intel® Core™ i7-10700 @ 2.90GHzx16
- OMNeT++ 5.6.2

$$NMSE(mean\ square\ error) = 10log_{10}(\frac{\sum_{t=1}^{T}|e(t)|^2}{\sum_{t=1}^{T}|r(t)|^2})$$

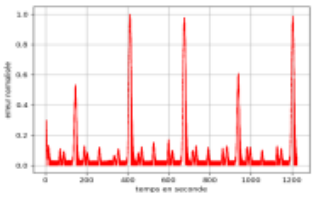




Protocole :
COPADRIVe (PRC) et PFE ECE - Challenge UTAC CERAM comparés, tous les deux basés sur ROS pour des projets avec convoi, COPADRIVe est un PRC apportant la communication sur OMNeT++.
Par l'utilisation des topics sur ROS, récupération des commande de vitesse par CAMs (Cooperative Awareness Messages) et les mesures de vitesses réelles.
La mesure de la vitesse réelle des véhicules et de la vitesse commandée seront intégrées par une formule d'erreur $e(t) = r(t) - y(t)$ (avec $r(t)$ la vitesse commandée et $y(t)$ la vitesse réelle qui) sera normalisée pour ensuite en obtenir l'erreur quadratique moyenne normalisée (NMSE) afin de déterminer quelle plateforme à un meilleure impact sur la trajectoire des véhicules. Une valeur faible de NMSE correspond à une plus grande précision.

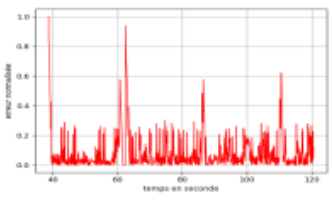
RÉSULTATS

COPADRIVe (PRC)



$NMSE_{COPA} = 19{,}2999$

PFE ECE



$NMSE_{PFE} = 34{,}5413$

Ces graphiques montrent l'erreur normalisée au cours du temps pour les deux projets. On peut remarquer que l'erreur est plus régulière pour le projet PFE, mais il arrive que des pics importants d'erreurs arrivent sur COPADRIVe.

DISCUSSIONS & PERSPECTIVE

- Dans la simulation, on obtient des erreurs de vitesse moins importants avec COPADRIVe.
- Les méthodes pour récupérer les données ne sont pas facilement accessibles, ce qui pourrait être intéressant dans le futur pour le développement d'une PRC de générer facilement les données pour analyse.
- Possiblement pour un projet PFE futur, il serait bienvenu de s'inspirer de COPADRIVe ou d'un projet récent inspiré de celui-ci, AuNa [3], pour développer une analyse de communication avec les DENMs (Decentralized Environmental Notification Messages) afin de regarder les interactions avec l'environnemment.

RÉFÉRENCES & BIBLIOGRAPHIE

[1] A. Koubaa. Robot Operating System (ROS) : The Complete Reference (Volume 6). Studies in Computational Intelligence. Springer International Publishing, 2021.
[2] Giovanni Nardini, Dario Sabella, Giovanni Stea, Purvi Thakkar, and Antonio Virdis. Simu5g–an omnet++ library for end-to-end performance evaluation of 5g networks, 2020.
[3] Harun Teper, Anggera Bayuwindra, Raphael Riebl, Ricardo Severino, Jian-Jia Chen, andKuan-Hsun Chen. Auna : Modularly integrated simulation framework for cooperative autonomous navigation. 07 2022.
[4] Bruno Vieira, Ricardo Severino, Enio Vasconcelos Filho, Anis Koubaa, and Eduardo Tovar. Copadrive - a realistic simulation framework for cooperative autonomous driving applications. In2019 IEEE International Conference on Connected Vehicles and Expo (ICCVE),pages 1–6, Nov 2019.

Fiche d'évaluation ''Entreprise'' à compléter par le Maître de stage

à insérer dans le rapport de stage

Fiche d'évaluation du stage
Cycle Ingénieur • 2ème année
2021/2022

Etudiant : ……Joseph Louville…….. Majeure : SE……..

Entreprise : ……ECE……...

Tuteur de stage : ……Naila Bouchemal, Jae Yun Jun Kim……..

☎ du tuteur : …… Email du tuteur : naila.bouchemal@ece.fr, jae-yun.jun-kim@ece.fr

	Possibilité d'accorder des ½ points	**Note**
Technique • Difficulté de l'étude • Résultats obtenus / respect du CDC • Qualité du travail (méthodologie, réalisation, dossier …) **Total Note technique**		 3/4 4/6 8/10 **/20**
Qualités humaines • Autonomie et sens des responsabilités • Esprit d'initiative et créativité • Intégration / sociabilité au sein de l'équipe • Comportement / Attitude **Total note Qualités Humaines**		 5/7 2/5 5/5 3/3 **/20**
TOTAL		**32/40**
Soit		16/20

Observations :
Très bon stage qui a permis l'avancement d'un projet important pour l'école.
Il faut améliorer certains points :
- Organisation, il faut être plus méticuleux et respecter plus soigneusement l'enchainement des tâches.
- Il ne faut pas hésiter à être force de proposition, et aller au delas des solution existantes (proposer de nouvelles solutions)

A ……Paris……, le ……09/09…….2022
Signature du Tuteur de stage
(obligatoire)

Cachet de l'entreprise d'accueil
(obligatoire)