RAPPORT FINAL

Celeste KOLOUSSA | Vincent ARIMON

SAE5–6

Préparation pour la course F1tenth

Sommaire

I. Introduction et Contexte

I.I Présentation du Projet

Le projet consiste à développer et intégrer des fonctionnalités avancées pour une voiture autonome pour une participation au concours F1Tenth. L'objectif principal est d'améliorer les capacités de la voiture en planification de trajectoire, dépassement des concurrents, évitement d'obstacle et optimisation de la vitesse.

I.II Contexte et Enjeux

Avec le développement rapide des technologies liées à la conduite autonome, la participation au concours F1Tenth constitue une occasion précieuse de perfectionner les systèmes embarqués dans un contexte de course compétitif. Ce projet s'inscrit dans une démarche de recherche et développement visant à améliorer les capacités d'interaction dynamique entre la voiture autonome et un environnement complexe, comprenant des obstacles statiques et dynamiques.

I.III Objectifs Généraux et Spécifiques

- Développer des algorithmes robustes de planification de trajectoire.
- Mettre en place des mécanismes efficaces de dépassement et d'optimisation de vitesse.
- Assurer une intégration harmonieuse avec les algorithmes existants pour une prise de décision cohérente.
- Collaborer avec un projet complémentaire visant à établir une communication entre la voiture autonome et des feux tricolores.

II. Cahier des Charges

II.I Description des Besoins

- Intégration de nouvelles fonctionnalités pour une gestion optimale des courses (planification, dépassement, vitesse).
- Développement en C/C++ et utilisation du système ROS pour l'orchestration logicielle.
- Collaboration interprojet pour gérer les interactions avec l'environnement extérieur via des communications HTTP ou l'analyse vidéo.

II.II Contraintes et Limitations

- Robustesse face à des scénarios variés et complexes, incluant des interactions avec d'autres voitures.
- Limitation des ressources matérielles (Jetson Nano) pour le développement et l'exécution des algorithmes.
- Nécessité d'intégrer les nouvelles fonctionnalités sans perturber les algorithmes existants.

II.III Analyse de l'État de l'Art

Les algorithmes d'autonomie pour les véhicules sont souvent focalisés sur des sous-systèmes spécifiques, comme la perception, la localisation ou la planification. Le projet F1Tenth se distingue par son approche intégrée dans un contexte de haute performance. Les principales références incluent :

- Les techniques de planification de trajectoire basées sur des approches de type A* et RRT (Rapidly-exploring Random Trees).

- Les algorithmes d'optimisation de vitesse comme le modèle PID ajusté aux scénarios dynamiques.

- Les systèmes ROS largement adoptés dans les projets robotiques

II.IV Indicateurs de Performance (KPI)

- Pourcentage de dépassements réussis sans collision.

- Amélioration du temps total sur une course complète par rapport à l'algorithme de base.

- Pourcentage d'obstacle éviter

III. Gestion de Projet

III.I Planification et Stratégie

La stratégie de développement repose sur une méthodologie itérative et incrémentale afin de garantir la livraison progressive des fonctionnalités et leur intégration dans le système global. Les étapes principales incluent :

- Analyse des besoins et spécifications : Comprendre en détail les fonctionnalités à développer, notamment la planification de trajectoire, le dépassement, et l'optimisation de la vitesse.

- Développement des modules fonctionnels : Développer et tester indépendamment chaque fonctionnalité.
- Intégration avec ROS : Assurer la compatibilité et l'orchestration des modules au sein du système ROS.

- Validation et tests sur la Jetson Nano : Effectuer des essais en conditions simulées et réelles pour affiner les algorithmes.

III.II Chiffrage et Estimation des Ressources

Les ressources nécessaires pour le projet sont estimées comme suit :

- Temps de développement :
    - Phase d'analyse : 1 semaines
    - Développement des modules : 7 semaines
    - Validation finale et documentation : 1 semaines

- Équipe :
    - 2 étudiants en logiciels expérimentés en C++

- Ressources matérielles :Jetson Nano et accessoires (Capteurs US, capteurs LiDAR).
    - Ordinateur de développement pour la connexion SSH.
    - Composant électronique
    - Machine de Fabrication/Assemblage de carte électronique

- Budget estimé :
    - Composant Électronique (ESP32): 10 €

III.III Planning Prévisionnel (Diagramme de GANTT)

Lors de la **deuxième partie de notre SAé, la SAé6**, nous avons **divisé notre temps** de travail de la manière suivante, afin d'optimiser **le développement, les tests et l'intégration des fonctionnalités** essentielles du projet.

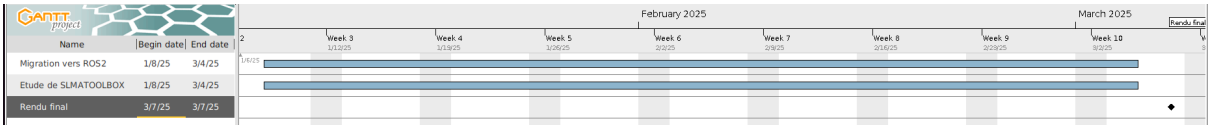

Figure 1 - GANTT PROJECT

IV. Implanter/Installer/Intégrer

IV.I Description de la solution réalisée

Nous avons réalisé une voiture autonome fonctionnelle qui peut utiliser un freinage d'urgence avec lidar ou capteurs US, qui est capable de circuler dans un circuit et d'adapter sa vitesse en fonction de la distance entre elle et le prochain obstacle.

IV.II Schéma de principe

Notre solution logicielle se présente comme ci-dessous :

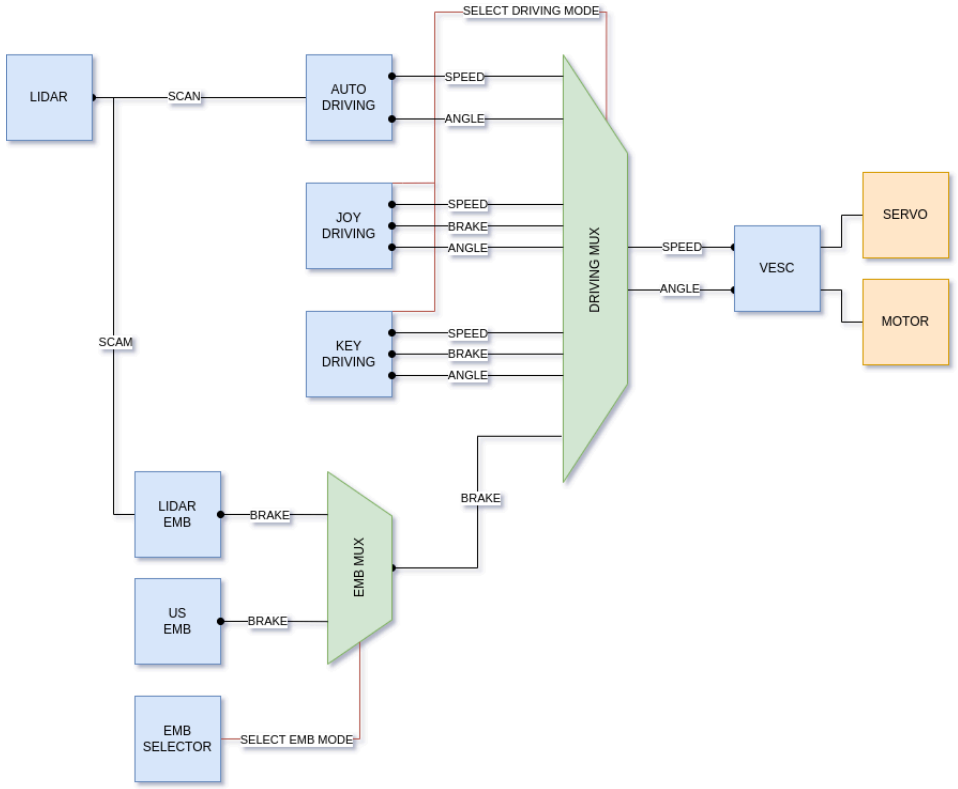

Figure 2 - Algorigramme

Voici une description détaillée de chaque nœud du système, en précisant leur rôle, leurs entrées et leurs sorties :

1. Nœud RPLIDAR (Package ROS RPLIDAR)

- Rôle :

Ce nœud récupère les données du capteur LIDAR, effectue le scan laser en temps réel et publie les informations sous forme de cloud de points.

- Entrées :
  - Données brutes provenant du capteur RPLIDAR connecté.
- Sorties :
  - /scan : Topic où sont publiées les données des scans laser
- Utilisation :
  - Les données publiées dans /scan sont utilisées par le nœud AUTO DRIVING pour la conduite autonome.

○ Elles sont également exploitées par le nœud EMB MUX pour la logique de freinage d'urgence via le lidar.

2. Nœud AUTO DRIVING

- Rôle :

Assure la conduite autonome en utilisant les données publiées sur le topic /scan par le nœud RPLIDAR.

- Entrées :
  - /scan : Topic contenant les données du LIDAR.
- Sorties :
  - VITESSE (SPEED) : Commande de vitesse pour le moteur.
  - ANGLE : Commande de direction pour le servo.
  - Ces sorties sont envoyées vers le DRIVING MUX.

3. Nœud JOY DRIVING

- Rôle :

Contrôle manuel du robot via une manette (joystick).

- Entrées :
  - /joy : Topic contenant les axes et boutons du joystick.
- Sorties :
  - VITESSE (SPEED) : Commande de vitesse pour le moteur.
  - ANGLE : Commande de direction pour le servo.
  - Ces sorties sont également envoyées vers le DRIVING MUX.

4. Nœud KEY DRIVING

- Rôle :

Gère le contrôle manuel via clavier pour piloter la vitesse et la direction du robot.

- Entrées :
  - Commandes clavier (interprétées par le programme).
- Sorties :
  - VITESSE (SPEED) : Commande de vitesse.
  - ANGLE : Commande de direction.
  - Ces commandes sont envoyées au DRIVING MUX.

5. Nœud DRIVING MUX

- Rôle :

Multiplexeur central pour sélectionner le mode de conduite actif (AUTO, JOY ou KEY) et envoyer les commandes finales aux actionneurs.

- Entrées :
  - VITESSE et ANGLE des trois modes :
    - AUTO DRIVING
    - JOY DRIVING
    - KEY DRIVING
  - Signal de sélection du mode de conduite.
- Sorties :
  - VITESSE : Commande finale pour le moteur.
  - ANGLE : Commande finale pour le servo.
  - Ces sorties sont envoyées au nœud VESC.

6. Nœud EMB MUX

- Rôle :

Combine les informations de détection d'obstacles pour activer le freinage d'urgence.

- Entrées :
  - LIDAR EMB : Données de détection d'obstacles via le topic /scan du nœud RPLIDAR.
  - US EMB : Données provenant des capteurs ultrasons.
  - EMB SELECTOR : Sélectionne la logique de freinage active.
- Sorties :
  - BRAKE : Signal de freinage envoyé au DRIVING MUX pour arrêter le robot.

7. Nœud VESC

- Rôle :

Exécute les commandes finales envoyées par le DRIVING MUX pour contrôler les actionneurs.

- Entrées :
  - VITESSE : Commande pour contrôler la vitesse du moteur.
  - ANGLE : Commande pour contrôler la direction via le servo.
- Sorties :
  - Contrôle du MOTEUR pour la vitesse.
  - Contrôle du SERVO pour la direction.

Résumé de l'Architecture

- Le nœud RPLIDAR collecte les données du capteur LIDAR et les publie dans le topic /scan.
- Les nœuds AUTO DRIVING, JOY DRIVING et KEY DRIVING génèrent des commandes de vitesse et angle selon le mode de conduite actif.
- Le DRIVING MUX sélectionne les commandes actives et les envoie au nœud VESC pour contrôler le moteur et le servo.
- Le EMB MUX gère la logique de freinage d'urgence en combinant les données LIDAR et ultrasons pour assurer la sécurité du robot.

Cette architecture modulaire permet d'assurer une gestion claire et efficace de la navigation, tout en offrant un mode d'arrêt d'urgence robuste.

IV.III Difficultés rencontrées, solutions apportées

- **Problème d'installation de ROS sur Jetson Nano**
    - Incompatibilité entre certaines versions de ROS et les pilotes GPU du Jetson.
    - **Solution** : Installation d'une version spécifique de ROS et adaptation des dépendances.
- **Alimentation instable du Jetson Nano**
    - La batterie initiale causait une mise en mode sécurisé de la carte.
    - **Solution** : Remplacement par une alimentation stabilisée et utilisation d'un régulateur de tension.
- **Configuration du contrôleur VESC**
    - Paramétrage complexe pour assurer un contrôle fluide du moteur.
    - **Solution** : Ajustement des valeurs PID et optimisation des temps de réponse.

V. Vérifier

V.I Protocole d'évaluation

Pour **évaluer notre solution**, nous avons réalisé des **tests sur circuit** en utilisant **différentes vitesses** afin d'analyser le comportement du véhicule autonome dans divers scénarios. Ces tests nous ont permis de **vérifier la précision de la trajectoire, la réactivité du freinage d'urgence et la fiabilité des capteurs** face aux obstacles.

V.II KPI

Nous avons évalué les performances de la voiture sur sa capacité à éviter les obstacles, à circuler dans un circuit et à freiner quand la voiture est trop proche d'un obstacle (freinage d'urgence).

V.III Comparaison avec les objectifs

Objectif atteint : Développement d'un système ROS fonctionnel et implémentation des algorithmes clés.
Objectif partiellement atteint : La transition complète vers **ROS2** est encore en cours.
Objectif à améliorer : L'intégration des feux de circulation dans la prise de décision du véhicule.

VI. Maintenir

VI.I Plan de maintenance, documentation, impact écologique

Plan de maintenance

- Mise à jour continue des packages ROS et des scripts Python.
- Tests réguliers des capteurs pour éviter les dérives de calibration.
- Surveillance de l'état de la batterie et des composants électroniques.

Documentation

- Utilisation de **GitLab** pour gérer les versions et assurer une traçabilité du développement.

Impact écologique

- Réduction de la consommation énergétique grâce à des composants à faible consommation.
- Utilisation de **matériaux recyclables** pour la structure du châssis.
- Optimisation du routage PCB pour minimiser l'utilisation de cuivre et de matériaux toxiques.

VI.II Évolution possible

- **Finalisation de la transition vers ROS2** pour améliorer les performances du système.
- **Intégration d'un système de perception avancée** basé sur une **caméra 3D** et un traitement d'image embarqué.
- Utilisation de **SLAM** et intégration de **path-planning.**

VII. Bilan du projet Conclusion

Le projet **F1Tenth** nous a permis de travailler sur des technologies avancées en **systèmes embarqués et robotique autonome**. Nous avons acquis des compétences en **programmation ROS, gestion de capteurs, conception électronique et algorithmie embarquée**. Malgré certaines **difficultés techniques**, nous avons pu développer un prototype fonctionnel capable de naviguer de manière autonome.

Ce projet nous a également appris l'importance de **l'optimisation des performances en temps réel**, ainsi que la **collaboration en équipe** sur un projet pluridisciplinaire. La transition vers **ROS2** reste un défi majeur, mais elle représente une étape cruciale pour améliorer la robustesse et la fiabilité du système.

Nous sommes convaincus que les futurs développements permettront de **rendre ce véhicule encore plus performant** et d'envisager des applications plus larges dans le domaine des véhicules autonomes.

VIII. Glossaire

- **ROS (Robot Operating System)** : Middleware open-source permettant la communication entre différents composants d'un système robotique via un modèle de publication/souscription.

- **ROS2** : Nouvelle version de ROS avec des améliorations en matière de performances, de communication en temps réel et de sécurité, utilisant le middleware DDS (Data Distribution Service).

- **SLAM (Simultaneous Localization and Mapping)** : Algorithme permettant à un véhicule autonome de se localiser tout en construisant une carte de son environnement en temps réel.

- **SLAM Toolbox** : Package ROS2 permettant la cartographie et la localisation pour améliorer la navigation des véhicules autonomes.

- **Jetson Nano** : Ordinateur embarqué de NVIDIA utilisé pour l'exécution des algorithmes de traitement d'image et d'intelligence artificielle dans les systèmes embarqués.

- **LiDAR (Light Detection and Ranging)** : Capteur utilisant des faisceaux laser pour mesurer les distances et créer une représentation en 3D de l'environnement du véhicule.

- **Ultrasonic Sensors (Capteurs ultrasoniques)** : Capteurs mesurant les distances à l'aide d'ondes sonores, souvent utilisés pour détecter les obstacles à courte portée.

- **VESC (Vedder Electronic Speed Controller)** : Contrôleur de moteur utilisé pour gérer la vitesse et l'accélération des moteurs électriques brushless.

- **Path Planning (Planification de trajectoire)** : Processus de calcul du chemin optimal pour un véhicule autonome en fonction des obstacles et des objectifs de navigation.

- **GitLab** : Plateforme de gestion de versions et de collaboration permettant le suivi du code source et la coordination des développements.

- **ESP32** : Microcontrôleur utilisé pour l'acquisition et le traitement des données des capteurs ultrasoniques avant leur transmission au Jetson Nano.

- **Middleware** : Logiciel intermédiaire facilitant la communication entre différentes applications ou composants d'un système informatique.

Summary

The main objective of the project is to create/program an autonomous car capable of avoiding obstacles. This project took place during our 2 Semester of our third year of Electrical engineering and industrial computing course. The project was offered only to the Embedded systems speciality and we worked on it for 6 months, from September to February. It's an academic project but also a business project since it can lead to the participation at the F1Tenth race, an international race of autonomous cars. We were supervised by Mr ZAHAF, our teacher of embedded systems at Fleuriaye Campus of the IT of Nantes. To conduct this project we had 2 chassis with a Jetson Nano(computer), a motor, a servo-motor, sensor, and also some lessons about a Middleware we had to use and learn named ROS1 (Robotic Operating System). To make the development easier we worked with VsCode, Kicad and Gitlab. During the first part of the project, we learnt how to use ROS1 to be able to understand the previous code given by our supervisor. Then we resolved some problems of logic in this code and upgraded it by adding an emergency break and a speed control law. Secondly we encountered a problem, the main sensor, a lidar, didn't allow us to make an emergency break for dynamic obstacles. To solve this, Céleste designed and programmed an Ultrasonic sensor board to make up for the lidar problem. In fact the lidar can’t detect close range obstacles but the Ultrasonic sensors can. After that we integrated it in our main application.
We encountered another problem, with the booting of the computer, it went on secure mode, a mode that forbids us to boot. After some research with our supervisor we detected the source of the problem. It was because we were powering it by a battery, but when the battery was low it caused it to go on secure mode. To solve it we changed the power supply use when the car is not on tracks. At the end of this first part we wrote a report to explain to our supervisor what we have done and what’s left to do. We also showed the result at the “Portes Ouvertes” of the campus. For the second part, we had to migrate the application from ROS1 to ROS2 and to study and use a package ROS2 named SLAMTOOLBOX. The objective of this is to augment the capacity of calculus with ROS2 and to create a map of the environment of the car to optimize its trajectory and speed also called path planning. The difficulties encountered are that ROS2 does not use the same tools as ROS1 and for SLAMTOOLBOX, our computer has some dependency problems for the libraries used. To conclude, right now we have an autonomous car working and capable of avoiding obstacles? Thanks to this project we learnt how to use ROS1 and ROS2 but also trained our computing skills, developed our knowledge of the lidar. It was a great experience in terms of Computing but also in human skills.