RAPPORT DE PROJET

Noms des participants au projet :

- Romain JOURNEAU
- Tom SOARES
- Tony VANG
- Sami TISSIR

Nom du projet : COVACIEL-AI

Enseignant référant : Alexandre FACCHIN

Etablissement : Lycée La Fayette Champagne s/Seine

Formation : BTS CIEL-IR session 2025

Lycée La Fayette
Champagne-sur-Seine • Fontaineroux

SOMMAIRE

1. PRÉSENTATION DU PROJET

Le projet **CoVACIEL** consiste à développer une voiture autonome capable de participer à un concours de conduite autonome. L'objectif principal est de concevoir un véhicule capable de naviguer de manière autonome, sans l'intervention humaine sur un circuit délimité en détectant les bords du parcours et en évitant les obstacles, tout en optimisant sa vitesse et son orientation. Pour cela, la voiture doit être capable de suivre un tracé précis, d'éviter des obstacles, et d'optimiser ses trajectoires et sa vitesse en temps réel. Le tout piloter par une intelligence artificielle.
Ce projet s'inscrit donc dans un cadre technologique et éducatif pour démontrer la capacité à développer un système autonome fiable et performant.

Le projet repose sur un développement en **Python objet** et **Arduino** avec des bibliothèques tel que **OpenCV** (pour le traitement d'image) et **TensorRT** (accélération du deep learning).

Les défis techniques abordés sont similaires à ceux rencontrés dans l'industrie :

- **Perception de l'environnement** grâce à des capteurs (vision, ultrasons).
- **Prise de décision en temps réel** pour ajuster la trajectoire et éviter les obstacles.
- **Optimisation des performances** pour allier vitesse et sécurité…

2. EXPRESSION DU BESOIN

2.1 Attentes du projet

Le projet **CoVACIEL-AI** a donc pour objectif principal de concevoir une voiture radiocommandée transformée en véhicule autonome capable de circuler sur un circuit sans intervention humaine. Le véhicule doit parcourir un circuit balisé de manière totalement autonome. Le système doit être capable de suivre un parcours défini par des lignes colorées tout en évitant des obstacles statiques.

Objectifs principaux :

- **Autonomie complète** : La voiture doit naviguer de manière autonome sur le circuit en utilisant ses capteurs ultrason et sa caméra.
- **Détection et suivi de trajectoire** : Utilisation de la vision par ordinateur et des capteurs ultrason (appréciation des distances) pour détecter les **murs colorées** (**vert** à **droite** et **rouge** à **gauche**) qui définissent les bordures du circuit et ajuster la trajectoire en conséquence.
- **Évitement d'obstacles** : La voiture doit détecter les obstacles en temp réel et ajuster sa trajectoire pour les contourner sans sortir du circuit ni entrer en collision avec les bords du circuit ou des obstacles quelconques.
- **Fusion de données** : Les informations des capteurs doivent être combinées pour une meilleure précision.
- **Optimisation de la vitesse** : Le système doit adapter la vitesse du véhicule en fonction de la configuration du circuit (virages, lignes droites, obstacles…).
- **Sécurité et stabilité** : La voiture doit être capable de s'arrêter en cas de problème et d'assurer une navigation fluide et stable.

Contraintes de réalisation :

Contraintes de développement (matériel) :

- Système embarqué :
  - Une **Jetson Nano**, utilisée pour le traitement d'image et l'intelligence artificielle.
  - **Une carte Arduino,** chargée de la gestion des capteurs et du contrôle moteur et directionnelle.
- Un ensemble de capteurs embarqué, comprenant :
  - **Caméra IMX219** pour l'analyse de l'environnement.
  - **Capteurs ultrasons HC-SR04** pour la détection d'obstacles.
  - Fourche optique, déjà intégrer dans la voiture pour mesurer la vitesse de la voiture en temp réel.

Contraintes qualité (conformité, délais…) :

- Le projet doit être **finalisé avant la course** organisée après la validation par le jury.
- **Conditions environnementales** :
  - Fonctionnement **en intérieur et extérieur couvert**.
  - Adaptation aux **variations de luminosité**.
  - Résistance aux **vibrations**.
  - Température de fonctionnement entre **10°C et 35°C**.
- **Contraintes techniques** :
  - Utilisation exclusive de la **Jetson Nano** comme unité de calcul.
  - Alimentation par **batterie LiPo**.
  - Communication **sans fil** pour le monitoring.
  - Développement en **Python**.
  - Respect des **interfaces matérielles imposées**.
- **Contraintes temporelles** :
  - **Durée du projet** : 4 mois.
  - **Trois revues de projet** planifiées.
  - **Démonstration finale** devant le jury.
  - Respect des **jalons du planning**.

Contraintes de fiabilité et de sécurité :

- Temps de réponse : Le système doit fonctionner en temps réel, avec un délai minimal entre la détection des bordures, des obstacles et l'ajustement de la trajectoire.
- Fiabilité : Le système doit être stable et robuste, sans défaillances majeures pendant la durée de la course.
- Autonomie : La voiture doit être capable de fonctionner sans intervention humaine pendant toute la durée du parcours.
- Stabilité : L'algorithme de contrôle doit assurer une conduite fluide, même en cas de changements de luminosité ou d'irrégularités du sol.
- Maintenance : Le système doit être conçu pour être facilement testable et modifiable en cas de problème.

Spécifications techniques :

- Performance :
  - Temps de réponse < 100 ms pour la détection.
  - Précision du suivi de ligne : < 5 cm d'écart.
  - Vitesse minimale : 15 km/h.
  - Détection d'obstacles fiable à 360°.
  - Autonomie : minimum 30 minutes.
- Sécurité
  - Arrêt automatique en cas de perte des repères.
  - Limitation de vitesse en fonction de la visibilité.
  - Système d'arrêt d'urgence en cas de problème critique.
  - Protection contre les surcharges électriques.
  - Monitoring constant des paramètres critiques.

Livrables attendus

À la fin du projet, la voiture devra être capable de réaliser un parcours complet du circuit sans intervention humaine, tout en respectant les contraintes techniques et sécuritaires. Les livrables incluent :

- Un prototype fonctionnel de la voiture autonome.
- Un code source documenté et optimisé.
- Une documentation technique complète sur le matériel et les logiciels utilisés.
- Un rapport de tests détaillant les performances du système.
- Une démonstration finale prouvant la fiabilité du système.

2.2 État de l'art (Projets similaires)

2.2.1 Google Car (Waymo) : ce qui se fait au niveau industriel

Le projet **Waymo**, initialement appelé **Google Car**, est l'un des projets les plus avancés en matière de véhicules autonomes. Lancé en **2009** par Google, ce projet visait à développer une voiture capable de circuler sans conducteur, en utilisant des capteurs avancés et des algorithmes d'intelligence artificielle.

Principes de fonctionnement

Waymo repose sur plusieurs technologies clés pour assurer une conduite autonome :

Capteurs avancés :

- Lidar : Un **scanner laser 3D** qui cartographie l'environnement en haute résolution.
- Caméras : **Détection** des feux de circulation, panneaux de signalisation et obstacles.
- Radar : **Suivi** des véhicules et objets en mouvement, même par mauvais temps.

Intelligence artificielle et algorithmes :

- Utilisation du **deep learning** et de l'apprentissage automatique pour identifier les éléments de la route.
- Prise de décision en temps réel grâce à des **algorithmes** de planification de trajectoire.
- Modélisation du comportement des piétons et autres véhicules.

Systèmes de communication et de contrôle :

- **GPS haute précision** couplé à des cartes HD pour un positionnement précis.
- Un **logiciel de pilotage** qui ajuste la vitesse, les changements de direction et l'évitement d'obstacles.

Comparaison avec CoVACIEL :

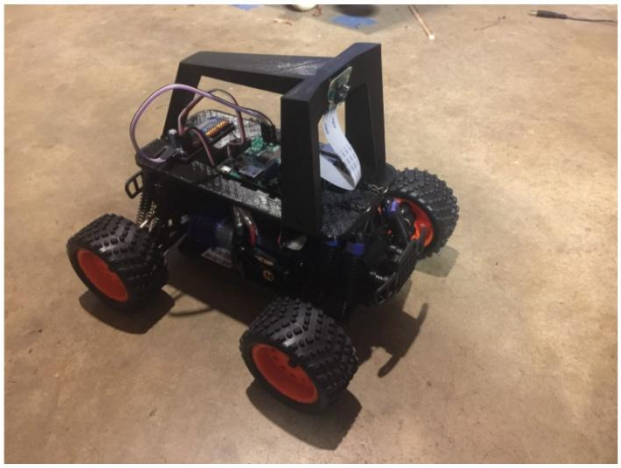
Bien que **CoVACIEL** soit un projet en modèle réduit, il partage des principes fondamentaux avec **Waymo** :

- Traitement de l'image pour analyser l'environnement.
- Fusion des données capteurs (caméra et capteurs embarqués).
- Navigation autonome, bien que sur une échelle plus petite.

Différences majeures :

- **Waymo** fonctionne en conditions **réelles** (rues, circulation, piétons), tandis que CoVACIEL est limité à une piste.
- **Waymo** utilise des **Lidar, radar et gps**, tandis que CoVACIEL repose uniquement sur la vision et ses capteurs ultrasons

2.2.2 Donkey car : ce qui se fait au niveau expérimentation académique/amateur



Donkey est une plateforme open source de conduite autonome pour voitures télécommandées, écrite en Python. Développée pour les amateurs et les étudiants, elle vise à accélérer l'expérimentation et à faciliter **les contributions de la communauté** . Elle prend en charge différents types de pilotage automatique, notamment les pilotes basés sur les réseaux neuronaux, la vision par ordinateur et le GPS. Elle est activement utilisée au lycée et à l'université pour l'apprentissage et la recherche.

Principes de fonctionnement

Donkey Car utilise des composants accessibles et un logiciel open-source basé sur **Python** et **TensorFlow** pour l'apprentissage automatique.

Matériel utilisé :

- Châssis RC : Une voiture radiocommandée comme base.
- Raspberry Pi : Unité de calcul principale pour exécuter les algorithmes.
- Caméra : Utilisée pour la vision et le suivi de trajectoire.
- Moteur et servo : Contrôlés par le Raspberry Pi pour ajuster la direction et la vitesse.

Technologies logicielles :

- Deep Learning (CNN - Réseaux de neurones convolutifs) pour apprendre les trajectoires.
- OpenCV pour le traitement d'image en temps réel.
- Contrôle PID pour ajuster les virages et stabiliser la trajectoire.
- Entraînement supervisé : L'utilisateur pilote d'abord la voiture en mode manuel pour collecter des données, puis l'IA apprend à reproduire ces comportements.

Comparaison avec CoVACIEL :

Il existe de nombreuses similitudes entre Donkey Car et CoVACIEL, ce dernier étant un projet jumeau :

- Navigation autonome.
- Utilisation d'une caméra embarquée pour détecter la trajectoire.
- Algorithmes de traitement d'image avec **OpenCV.**
- Utilisation d'un réseau de neurones pour l'apprentissage.

Différences majeures :

- **Donkey Car** est conçu pour une approche générique, alors que **CoVACIEL** est optimisé pour une course spécifique.
- **Donkey Car** supporte plusieurs types de capteurs (GPS, IMU, caméra, etc.), alors que CoVACIEL est uniquement basé sur la vision (caméra) et les capteurs ultrasons.

2.2.3 F1TENTH : ce qui se fait au niveau compétitif académique

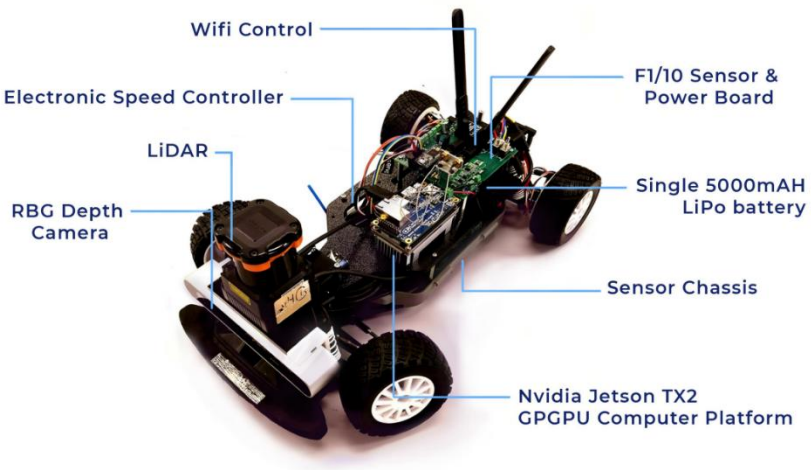
F1TENTH est une **compétition internationale** de voitures autonomes en modèle réduit (1/10e de la taille réelle), inspirée par la Formule 1. Les équipes universitaires du monde entier participent à cette compétition pour développer des voitures autonomes performantes alliant vitesse, précision et réactivité.

Principes de fonctionnement :

La course F1TENTH repose sur des voitures télécommandées modifiées, dotées de capteurs et d'algorithmes avancés.

Matériel utilisé :

- Châssis RC 1/10 de type buggy ou voiture de course.
- Lidar : Cartographie l'environnement pour la détection des obstacles.
- Caméra : Vision par ordinateur pour détecter la piste et les virages.
- Ordinateur embarqué NVIDIA Jetson pour le calcul et l'exécution de l'IA.
- IMU (Inertial Measurement Unit) pour estimer l'orientation et la vitesse.



Technologies logicielles :

- SLAM (Simultaneous Localization and Mapping) pour cartographier la piste.
- Path Planning (planification de trajectoire) avec algorithmes de graphes et optimisation.
- Deep Reinforcement Learning pour affiner les stratégies de conduite.

- ROS (Robot Operating System) pour la communication entre les capteurs et les algorithmes.

Comparaison avec CoVACIEL :

Comme CoVACIEL, F1TENTH est un projet de course autonome basé sur la vision par ordinateur.

- Utilisation de caméras et capteurs pour détecter la piste et les obstacles.
- Course autonome avec optimisation de la trajectoire.
- Approche expérimentale universitaire.

Différences majeures :

- **F1TENTH** utilise un Lidar et une caméra, tandis que **CoVACIEL** est uniquement basé sur la vision et les capteurs ultrasons.

3. RÉPARTITION DES TÂCHES

Les tâches ont été réparties entre les membres de l'équipe en fonction des besoins du projet :

3.1 MainController (Coordination des fonctionnalités)

Cette tâche est réalisée par **Tom SOARES,** l'objectif principal est, la mise en place :

- D'une fusion des données
- De la coordination globale
- De l'interface utilisateur
- De la performance globale

3.2 VisionSystem (Traitement d'image et navigation visuelle)

Cette tâche est réalisée par **Romain JOURNEAU,** l'objectif principal est, la mise en place :

- Du traitement d'image
- De la détection de lignes/obstacles
- Des contraintes en temps réel sur la vision
- De la navigation visuelle

3.3 SensorSystem (Gestion des capteurs et sécurité)

Cette tâche est réalisée par **Sami TISSIR,** l'objectif principal est, la mise en place :

- Du la gestion des capteurs ultrason
- De la sécurité du système
- De la détection des obstacles à proximité
- Du temps de réponse, de manière à qu'il soit le plus rapide possible

3.4 PilotSystem (Contrôle moteur et direction)

Cette tâche est réalisée par **Tony VANG,** l'objectif principal est, la mise en place :

- Du contrôle des mouvement du système
- De la navigation physique du système
- De la stabilité
- Des performances mécaniques

4. RECONFIGURATION DU PROJET

4.1 Stratégie initiale

Initialement nous devions pouvoir contrôler le moteur de la propulsion et le servomoteur directionnel, puis calculer la vitesse de la voiture grâce aux bornes de la **Jetson Nano** (Okdo nano c100). Plusieurs tests ont été effectué mais aucun n'a été concluant : incompatibilité de la librairie **Jetson.GPIO**, utilisation du **file-system** (avec l'aide du professeur) fonctionnel mais non optimale pour l'utilisation des moteurs, l'envoie et la réception des données (sans prendre en compte le fonctionnement en simultané). Continuer sur cette voie aurait été trop chronophage et n'aurait probablement pas aboutie au fonctionnement complet du projet…

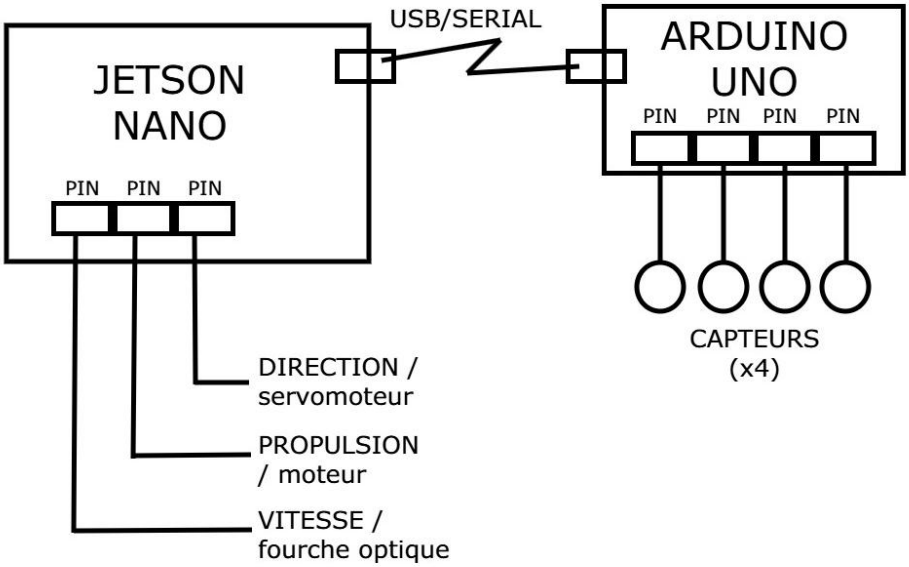


Figure 1 : STRATÉGIE INITIALE

4.2 Stratégie sélectionnée

Après s'être concerté avec le professeur, nous avions décidé de changer de méthode pour le contrôle de la voiture et le retour de la vitesse réelle.
La gestion du **PilotSystem** se fera donc sur **l'arduino UNO** avec la gestion du **SensorSystem**, elle communiquera avec la **Jetson Nano** (Okdo Nano c100) via USB/Serial.
Nous utilisons une vitesse de transmission de **115200 bauds** permettant environ 288 échanges d'informations par seconde avec la norme que l'on a imposé sachant que l'on veut au moins 100 échanges par seconde.

Format d'une valeur : **XX.XX**; soit 6 caratère/octets. On a besoin de 6 valeurs (4 capteurs de distance, vitesse réelle et le temps) pour envoyer à la Jetson Nano et traiter les informations, donc **6*6=36** mais on rajoute le caractère de retour à la ligne pour séparer les données envoyées et reçues, donc **36+1=37.**

On rajoute 2 valeurs (vitesse ciblée et direction ciblée) à recevoir pour éxécuter les actions, donc **6*2=12** idem pour le retour à la ligne **12+1=13.**
Cela nous fait **37+13=50 octets** par échanges soit **400 bits (50*8=400).**

On calcule le temps que met un seul échange avec 115200 bauds :
400/115200≃0.00347 secondes soit **3.47 milisecondes.**
On finit avec ça pour savoir le nombre d'échanges possible en 1 seconde entière : **1/0.00347≃288** échanges par seconde.

Le moteur ainsi que le servomoteur sont donc désormais branchés sur des bornes **PWM** tandis que la fourche est branché sur une borne analogique (provisoir? car en cours de test).
Pour que le **SensorSystem** et le **PilotSystem** fonctionne en simultané sur l'arduino, il faut donc que toute les classes soient codé dans le même fichier et qu'elles n'interfèrent pas entre elles.
Le code devra donc (dans le **loop()**):

1. Lire les consignes envoyées par le **MainController** (Jetson --[13 octets]--> Arduino)
2. Exécuter la vitesse et la direction ciblées (PilotSystem)
3. Lire les valeurs des 4 capteurs de distance, le temps (SensorSystem) et la vitesse réelle (PilotSystem)
4. Retourner les données du **SensorSystem** et du **PilotSystem** (Arduino --[37 octets]--> Jetson)

Et cela toutes les **100 milisecondes.**

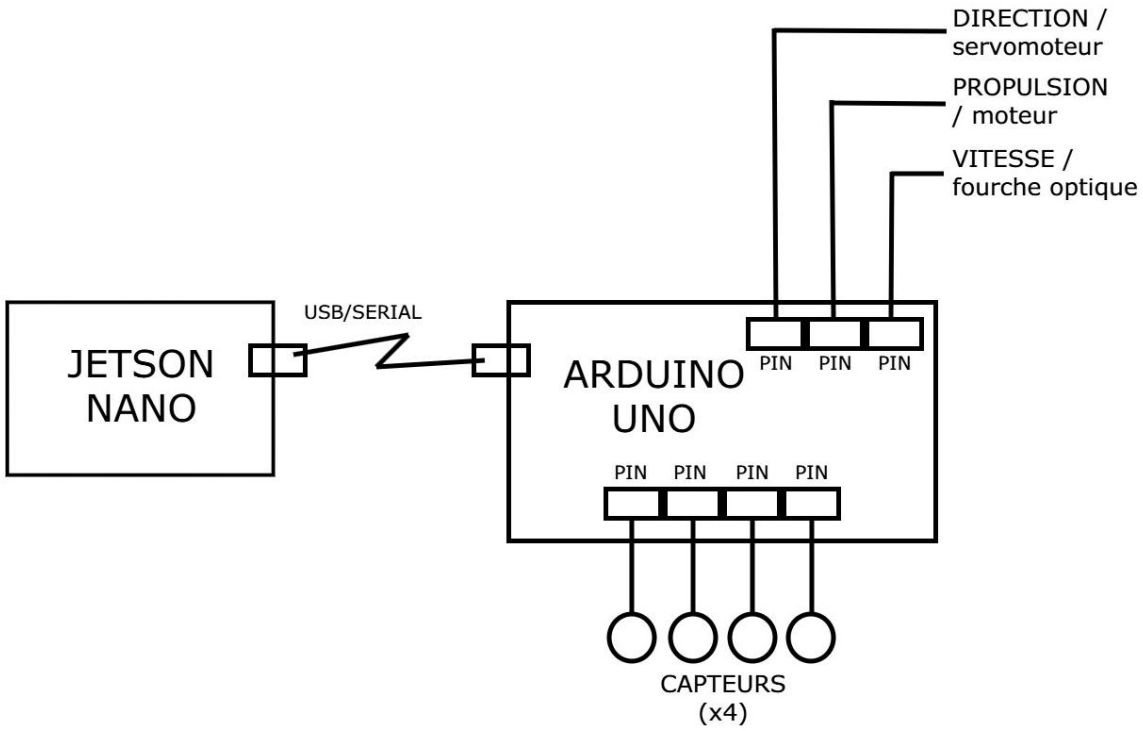


Figure 2 : STATÉGIE SÉLECTIONNÉE

5. DIAGRAMMES

6. PROBLÉMATIQUE DE PHYSIQUE

6.1 Problématique

« Comment créer un asservissement de vitesse tout en utilisant les informations de vitesse fournies par la fourche optique et le moteur en tant qu'actionneur ? »

6.2 Plan d'action

7. CONCLUSION COMMUNE