Projet CannonBall RC

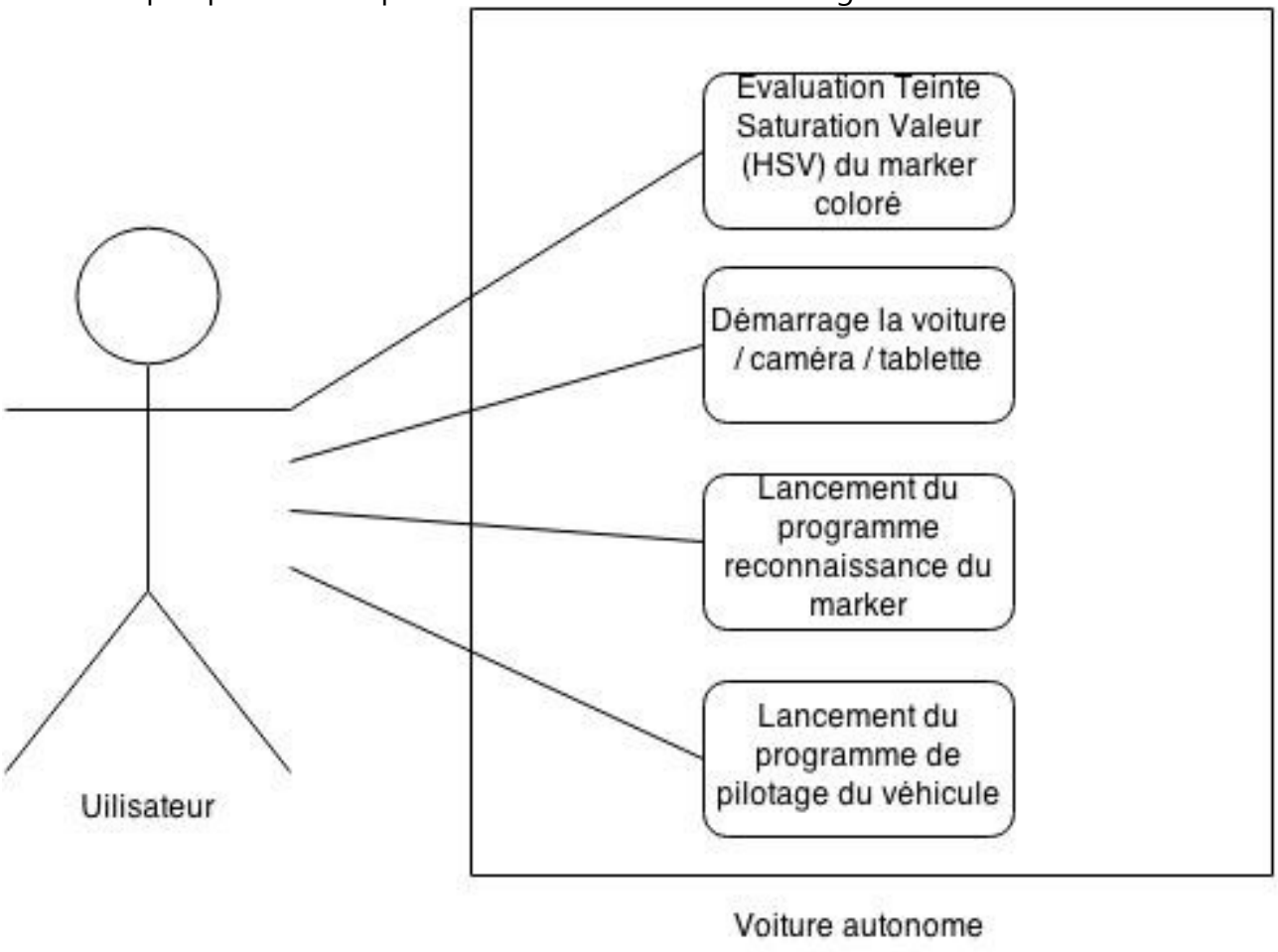
Table des matières

Contexte et domaine d'application ciblée

Le but de ce projet est la réalisation d'une voiture de course modèle réduit autonome capable de parcourir un circuit par identification de markers. Une des finalités est de faire un convoi de voitures qui maintiendraient une distance (de sécurité) entre elles.

Cahier des charges résumé

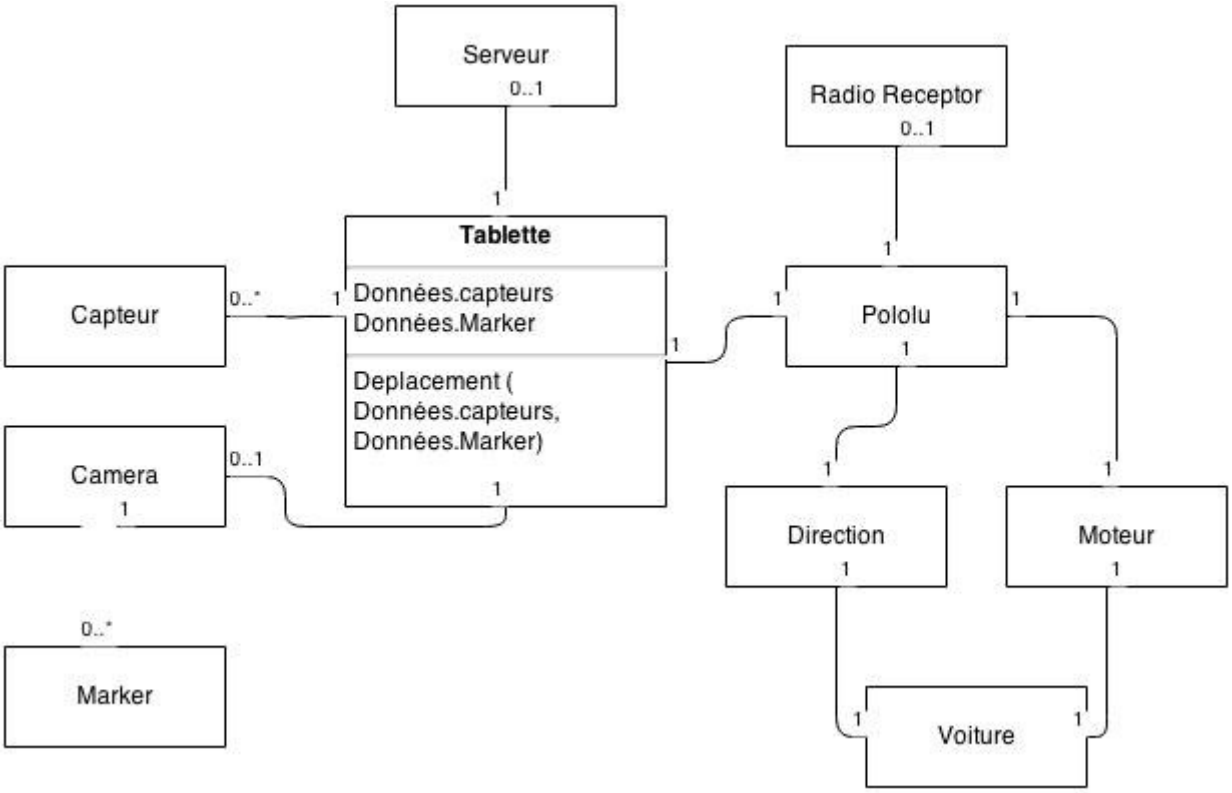
Le véhicule doit être capable de se déplacer à l'aide de marker,s pour cela, l'utilisateur doit effectuer quelques tâches qui sont schématisées dans le diagramme suivant :



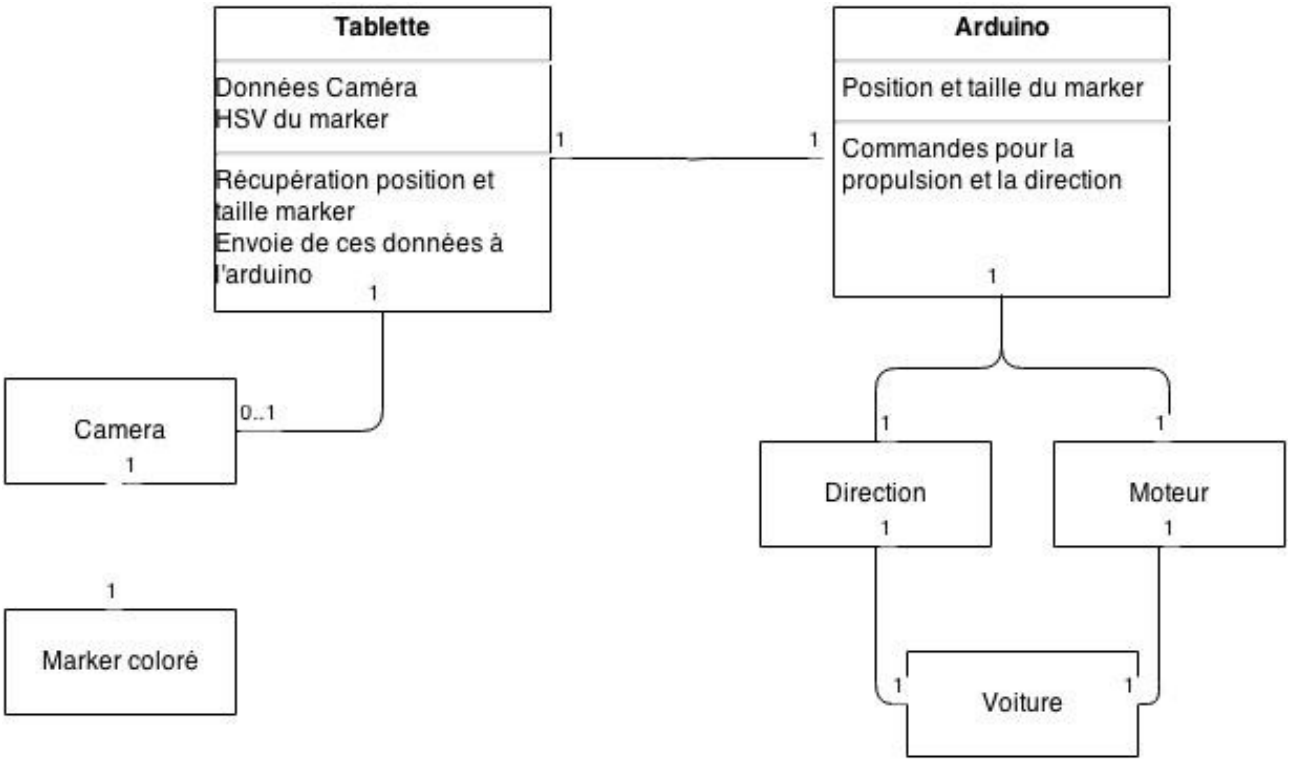
Tout ce que la voiture CannonBall nécessite comme énergie est fournie par une batterie lipo. L'utilisateur a juste à lancer le programme de reconnaissance de marker sur une tablette préalablement chargée et placer son marker coloré devant la caméra pour que la voiture agisse de manière autonome.

Achitectures implémentées

Premièrement, nous avions pensé à une architecture comportant plusieurs blocs inintéressants. Elle comprenait une partie Marker + Camera + Tablette, s'occupant de la détection , de l'analyse et du tracking des markers. Ensuite, une autre partie comprenant Le Pololu + les moteurs de direction et de propulsion + la voiture + la tablette, pour s'occuper du déplacement et du contrôle du véhicule. Enfin deux autres modules y étaient ajoutés : le serveur + les capteurs qui serviraient à envoyer des données sur un serveur via MQTT et aussi à optimiser la trajectoire du véhicule grâce à ces données. Enfin un module de Réception Radio couplé au Pololu, pour envoyer des ordres au véhicule, comme l'arrêt d'urgence.
Voici un diagramme présentant succinctement cette architecture :



Finalement, comme dans la plupart des cas, on fait face à des problèmes, et ce qui est prévu peut changer. Nous avons donc remplacé le Pololu par un Arduino dont la fonction reste globalement la même. L'usage des capteurs et du Radio Contrôle fut abandonné.
Voici un diagramme représentant l'architecture utilisée :



Composants logiciels et matériels utilisés

Matériel utilisé :

Tout d'abord, la base de notre montage, la voiture RC Kyosho Scorpion XXL VE équipéé de deux servo pour la direction et la propulsion. Elle contient un module de réception 2,4Ghz

-

- Tablette Lenovo Thinkpad2 sous Windows pour lancer les programmes d'analyse des markers et de contrôle du véhicule, choisi pour sa légèreté et sa finesse et son encombrement réduit.

- Arduino Uno pour le contrôle du véhicule via les servo-moteurs, Il pourra aussi recevoir les données du récepteur radio.

- Pololu maestro 6-channel servo controller : ancienne solution de contrôle du véhicule

- Caméra c920 Logitech filmant en HD pour une bonne précision de l'image qui est encodé en h264 matériellement pour la légèreté du flux.

Logiciel :

- OS :
  Windows qui était installé sur la tablette non compatible Linux (GPU)
- IDE :
  Visual Studio pour développer sous Windows.
  Pololu Maestro controller pour le premiers essai de contrôle.
  Arduino IDE pour la version final du controller.

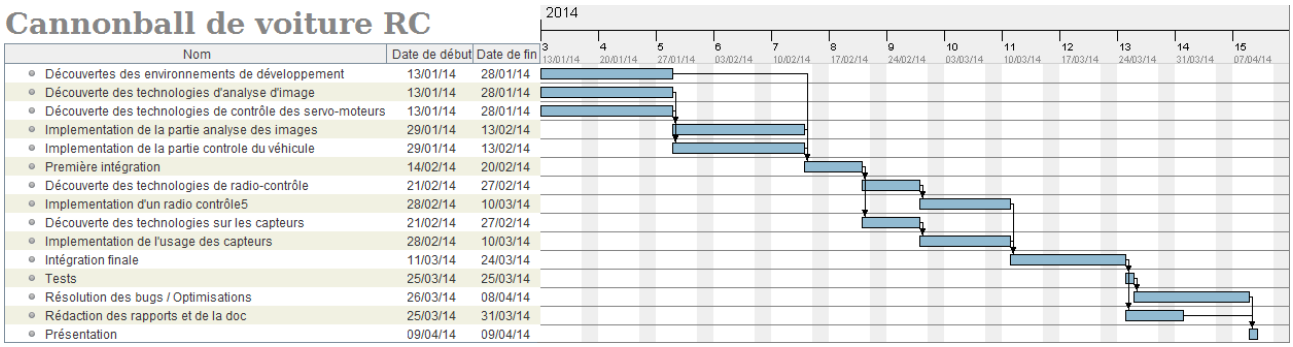
Technologies utilisées :

- Opencv (c++) pour la détection et la traking de markers
- Arduino pour le contrôle du véhicule
- Communication série pour faire communiquer les deux parties, contrôle et détection de markers, entre elles

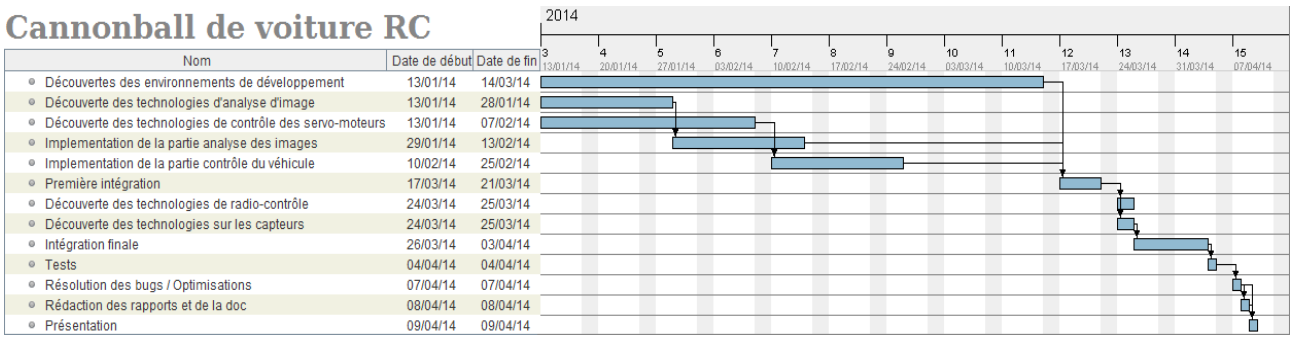
Gestion du projet (etapes, taches, temps, ...)

Tout d'abord nous avons imaginé une répartition idéale des tâches et du temps en deux grandes parties à savoir : la détection de markers et le contrôle du véhicule. Après la phase de développement des ces deux parties, il y aurait eu une mise en commun pour les intégrer ensembles afin de rendre le prototype fonctionnel. Une fois ce prototype achevé, les parties sur son optimisation auraient été abordées. Des parties telles que le contrôle de la voiture par Radio Contrôle, pour effectuer des arrêts d'urgences ; ou encore une partie sur les capteurs qui aurait permis un meilleur contrôle de la vitesse, des directions et de l'accélération. Enfin, une phase d'intégration finale aurait été nécessaire, suivie d'une phase de test, de debug et d'optimisation. Enfin, la documentation aurait pu être rédigée sereinement avant la soutenance.

Voici un diagramme de Gantt illustrant cette planification de projet :



Finalement, comme c'est souvent le cas, cette planification ne fut pas respectée, et suite a des problèmes auxquels nous avant du faire face, (problèmes développés au point suivant) le planning s'est vu quelque peu modifié, voici un diagramme illustrant ce que fut la planification réelle du projet :



Problèmes rencontrés et les solutions élaborées

Problème 1 : installation des librairies ArUco nécessaires à la reconnaissance des markers

Nous ne sommes pas parvenu à éditer correctement les librairies sous Windows (très peu de documentation).

--> solution : projet openCV trackant un marker coloré, et abandon de l'analyse de marker sous forme de Qrcode.

Problème 2 : services du Pololu insuffisants pour contrôler efficacement les moteurs du véhicules.

--> solution : récupération d'un Arduino, ce qui fut une bonne opération au vu de l'extrême richesse de la communauté.

Problème 3 : comment récupérer la position ainsi que la taille de l'objet à tracker ?

--> solution : utilisation de la fonction cvMoments(imgThresh, moments, 1);
Cette fonction calcule les moments spatiaux et centraux d'une image binarisée (ici imgTresh) jusqu'au troisième ordre. On peut ainsi obtenir les coordonnées (dans le repère de la caméra) du marker coloré que l'on souhaite tracker, mais aussi la taille de l'objet en question.

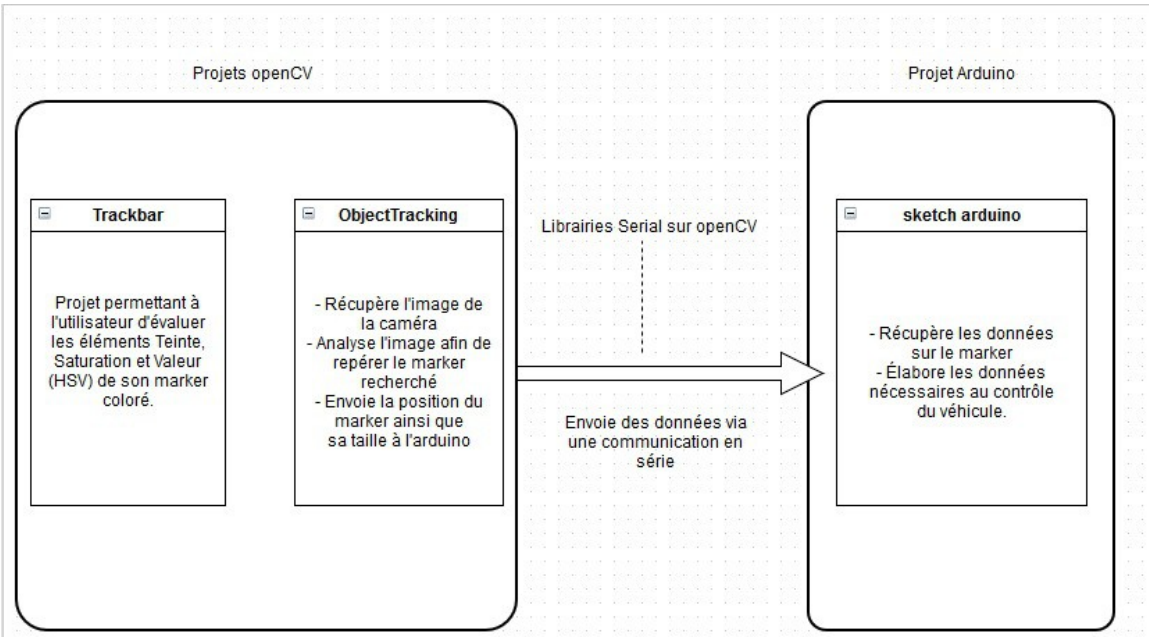
Problème 4 : intégration des parties "détection marker" et "pilotage véhicule".

Nous ne pouvions pas envoyer la position et la taille du marker nécessaires aux algorithmes de contrôle du véhicule sur l'Arduino.

--> solution : ajout de la librairie serial, permettant de transmettre des informations après les avoir préalablement découpées en plusieurs morceaux de taille fixe. Cette méthode permet l'envoie ainsi que la réception de données sur l'Arduino via son port Serial. La communication en série nous a permis d'établir un dialogue à sens unique ici entre l'analyse de la caméra et les moteurs (propulsion et direction).

Conclusion

Au final notre projet se divise en trois parties, dont une bien distincte :



Architecture Système

L'utilisateur détermine les valeurs TSV (Teinte Saturation Valeur) correspondantes au marker coloré, puis rentre ces données dans ObjectTracking. Il ne reste plus qu'à lancer le projet openCV ObjectTracking,, et à placer le marker dans le champs de vision de la caméra.
Ainsi, le véhicule, piloté par l'arduino grâce aux données envoyée par ObjectTracking, suivra le marker en gérant sa vitesse et sa direction de manière autonome.

Évolution et perspectives :

L'année prochaine, lors qu'un groupe aura repris le projet, il pourra être judicieux d'implémenter la reconnaissance de markers de type Qrcode, pour cela, la bibliothèque ARuco est tout indiquée. L'avantage de ces markers et d'être facilement détectables (contraste entre le blanc et le noir) et identifiables, cela évite tout parasitage.
Il serait plus aisée de travailler sous Linux pour pouvoir ajouter des bibliothèques plus aisément tout en gardant un tablette fine. Ici nous avons une tablette qui a une taille qui rentre parfaitement dans le châssis de la voiture, à 5 mm près,
L'utilisation des capteurs de la tablette pourra aussi être une perspective raisonnable, pour améliorer certains point comme la sécurité du matériel et des personnes. Grâce a ces capteurs, une mesure de la vitesse pourra être faite ainsi qu'une mesure de l'accélération pour contrôler ou éviter le dérapage.