Projet CannonBall

Réalisation d'une voiture de course modèle réduit autonome capable de parcourir un circuit par identification de markers.

Jules Legros & Benoit Perruche

Sommaire

- Contexte
- Objectifs du projet
- Gestion du projet
- Conclusion

Contexte

- **Matériel utilisé :**
  - Voiture RC Kyosho Scorpion XXL VE
  - Tablette Lenovo Thinkpad2
  - Arduino Uno
  - Caméra c920 Logitech
- **Logiciel :**
  - OS : Windows
  - IDE : VisualStudio et Arduino IDE
- **Technologies utilisées :**
  - OpenCV (C++)
  - Arduino
  - Communication en série

Objectif 1 : Autonomie voiture

- Autonomie énergétique
- Tous les composants nécessaires au projet sont sur le véhicule

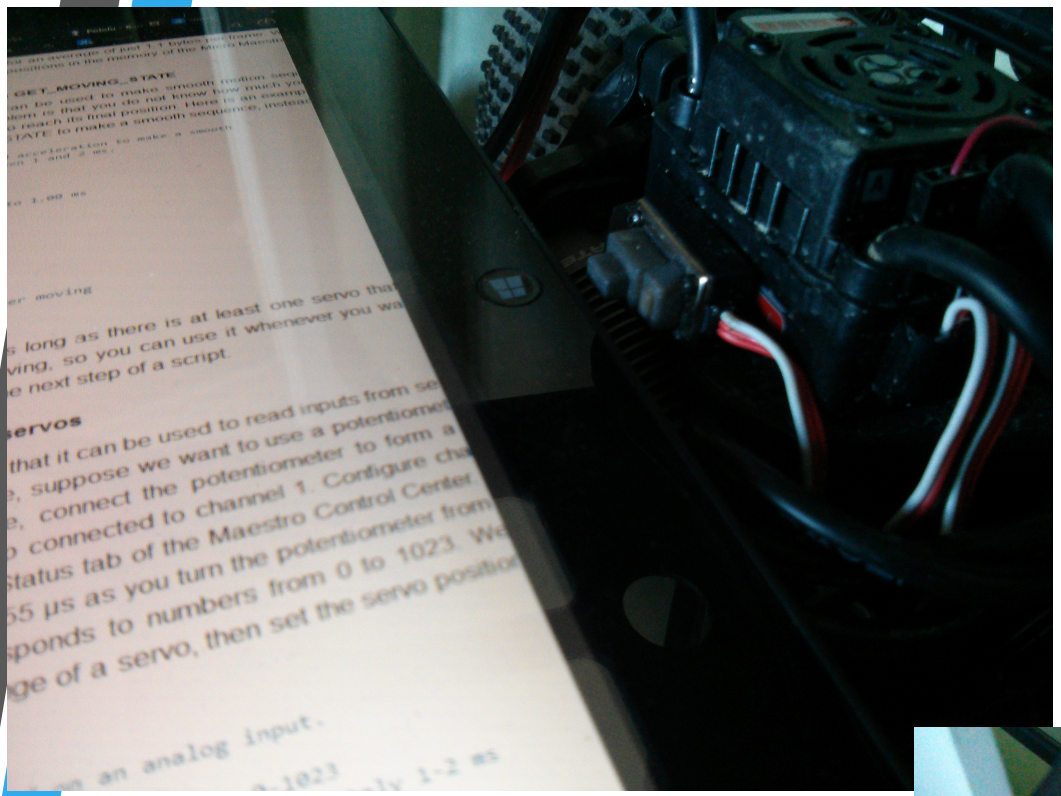
Problèmes et solutions apportées

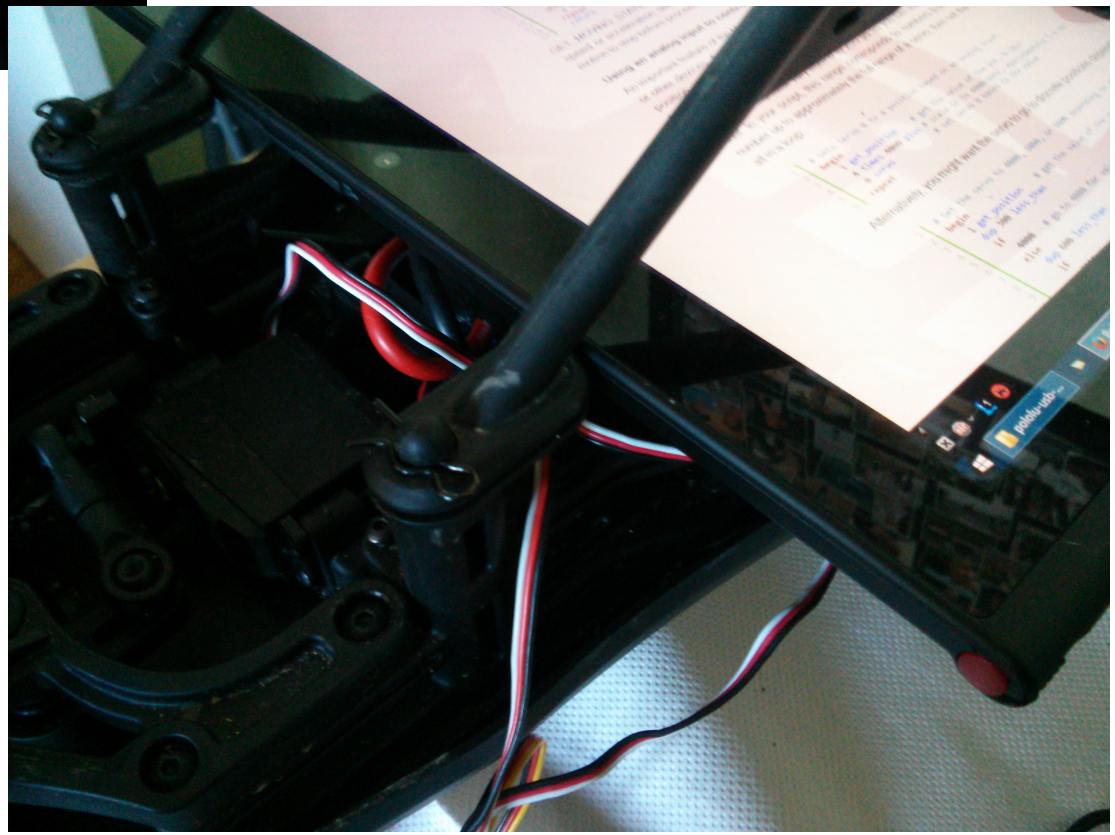
- Protection tablette contre les chocs

➢ Réalisation d'un boîtier de protection qui s'encastre dans le véhicule

- Amortissement des chocs subis par la caméra

➢ Support anti-vibrations


long as there is at least one servo tha
ving, so you can use it whenever you wa
e next step of a script.
servos
that it can be used to read inputs from se
, suppose we want to use a potentiome
, connect the potentiometer to form a
connected to channel 1. Configure cha
Status tab of the Maestro Control Center
55 µs as you turn the potentiometer from
sponds to numbers from 0 to 1023. We
ge of a servo, then set the servo positio
an analog input.



Objectif 2 : Reconnaissance de markers

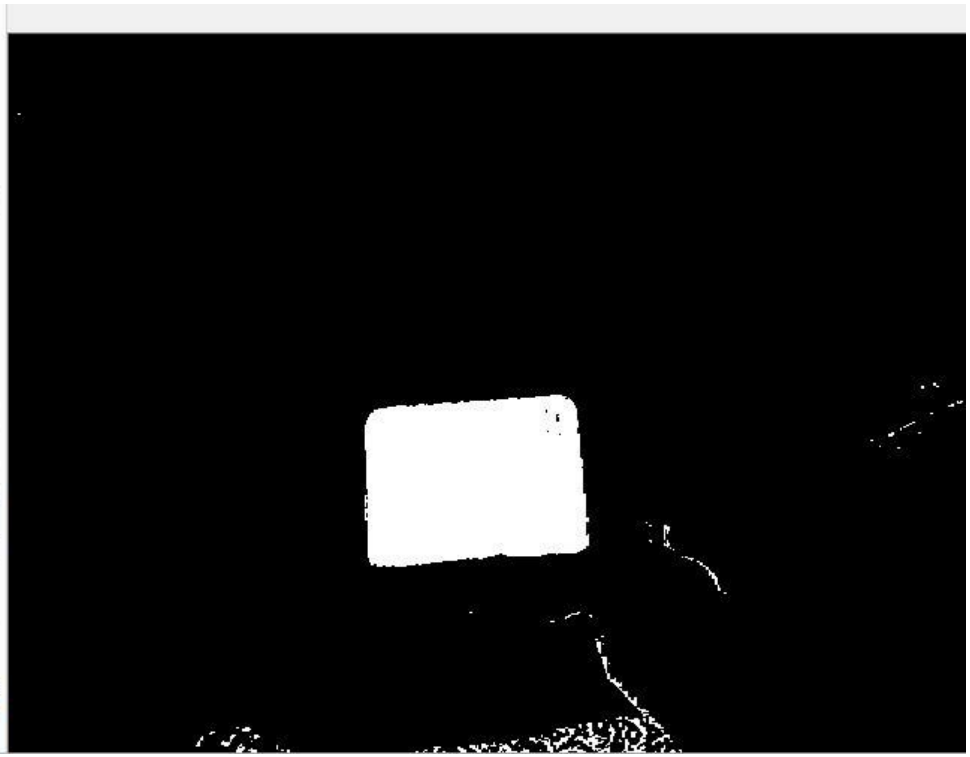
- Voiture capable de parcourir un circuit balisé par des markers

Problèmes et solutions apportées

- Impossibilité d'éditer correctement les librairies ArUco sous Windows (très peu de documentation)
  - projet openCV trackant un marker coloré

- Comment récupérer la position ainsi que la taille de l'objet à tracker ?
  - Fonction `cvMoments(imgThresh, moments, 1);`

tomatique
Variables locales
Espion 1
Pile des appels
Points d'ar



Objectif 3 : Algorithme de déplacement du véhicule

- Réalisation d’un algorithme de pilotage du véhicule (contrôle trajectoire, maîtrise de l'accélération, …)

Problèmes et solutions apportées

- Services du Pololu insuffisants pour contrôler efficacement les moteurs du véhicules
  - Récupération d'un Arduino (richesse de la communauté)

- Algorithmes de déplacement du véhicule
  - position de l'objet : direction
  - taille de l'objet : propulsion

```
position (284,443), taille (7149.00)
position (283,445), taille (6991.00)
position (288,442), taille (7014.00)
position (293,435), taille (7065.00)
position (302,422), taille (7199.00)
position (315,403), taille (7412.00)
position (350,352), taille (8248.00)
```

```
if (pos<641 && pos>-1) { // pour eviter les erreurs de positionnement
int posR = (int) pos * 90 / 640 + 45 ;  //on a une position en fonction de la taille de l'écran (640)
direction.write(posR);                  // l'angle de braquage va de 45 a 135 (range=90)
}

if (size<11500 && size>1500) {  //pour ne pas que la voiture aille trop vite

    // on a une vitesse en fonction de la taille de l'objet (plus il est gros, donc proche, moins elle est elevee)
    // on fixe un angle de vit_max comme vitesse max et 1500 comme valeure min de taille d'objet pour avancer a la vitesse max
    // avec une taille de 11500, la vitesse est nulle (objet trop pres) l'angle est de 90 (on entre dans la boucle suivante)

    int puiss = (int) (size-1500)/666 + vit_max;
    propulsion.write(puiss);

}else if (size >=11500 && size <=1500) { // objet tres pres ou objet perdu (ou trop loin...)

    propulsion.write(90);

}
```

Intégration des objectifs

- Intégration des parties "détection marker" et "pilotage véhicule"

Problèmes et solutions proposées

- Comment envoyer la position et la taille du marker nécessaires aux algorithmes de contrôle du véhicule sur l'Arduino ?

➢ Communication en série (via port serial de l'arduino)

```
//envoie de la position
LSB = posX & 0xff;
MSB = (posX >> 8) & 0xff;

arduin->WriteData(&MSB, 1);
arduin->WriteData(&LSB, 1);

//envoie de la taille
LSB = (int)area & 0xff; //printf("%u\n", LSB);
MSB = ((int)area >> 8) & 0xff; //printf("%u\n", MSB);

arduin->WriteData(&MSB, 1);
arduin->WriteData(&LSB, 1);
```

Objectifs non réalisés

- Évaluation des performances
- Infrastructure de collecte des paramètres de conduite du véhicule
- Algorithme de consensus permettant à plusieurs véhicules de rouler en convoi
  - Placer un marker coloré au dos du véhicules précédent

Gestion du projet
Plannification VS Réalité

Gestion du projet
Plannification VS Réalité

Rendu final

3 étapes :

- L'utilisateur détermine les valeurs TSV (Teinte Saturation Valeur) correspondantes au marker coloré
- Rentre ces données dans le projet ObjectTracking sur la tablette
- Place le marker dans le champs de vision de la caméra.

Évolution et perspectives

- Reconnaissance de markers de type Qrcode (bibliothèque ArUco)
- Utilisation des capteurs de la tablette

Avez-vous des questions ?

Démo