Rapport final de projet

Année scolaire 2024-2025

NXP Cup

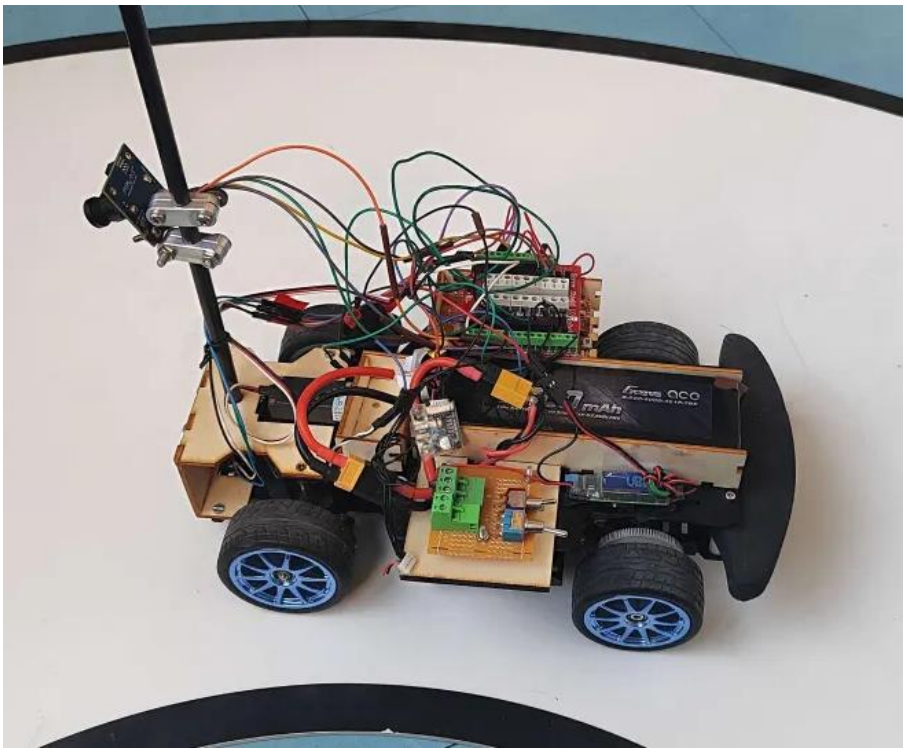


Figure 1 - Image du Robot

Etudiants : TIBAUDO Romain

Et

WARTSKI NARANJO Daniel

Ecole Polytechnique Universitaire de Nice Sophia-Antipolis, Département électronique
1645 route des Lucioles, Parc de Sophia Antipolis, 06410 BIOT

Sommaire

I. Introduction

La NXP Cup est une compétition étudiante internationale organisée par NXP Semiconductors. Elle a pour objectif de concevoir et programmer une voiture autonome capable de naviguer sur un circuit inconnu, balisé par des lignes noires délimitant la trajectoire. La voiture doit effectuer un tour de piste aussi rapidement que possible, puis franchir la ligne d'arrivée, où elle doit ralentir visiblement et s'arrêter à une distance précise de 10 cm d'un obstacle.

Dans ce cadre, notre groupe reprend le travail réalisé par Guillaume RIO et Oumnia BHIKIR, avec pour objectif principal d'optimiser les performances de leur véhicule afin qu'il parcoure le circuit plus rapidement. En complément, nous visons à améliorer le code en y ajoutant la détection de la ligne d'arrivée ainsi que l'intégration du capteur à ultrasons pour garantir un arrêt précis devant l'obstacle final.

Cahier des charges :

- **Détection de la piste :** Le robot doit identifier les lignes noires du circuit pour calculer l'erreur de position et ajuster sa trajectoire en temps réel.
- **Détection de la ligne d'arrivée :** La voiture doit reconnaître correctement la ligne d'arrivée afin de réduire sa vitesse de manière visible.
- **Détection de l'obstacle :** Le robot doit mesurer la distance le séparant de l'obstacle final à l'aide d'un capteur, et s'arrêter précisément à 10 cm de celui-ci.

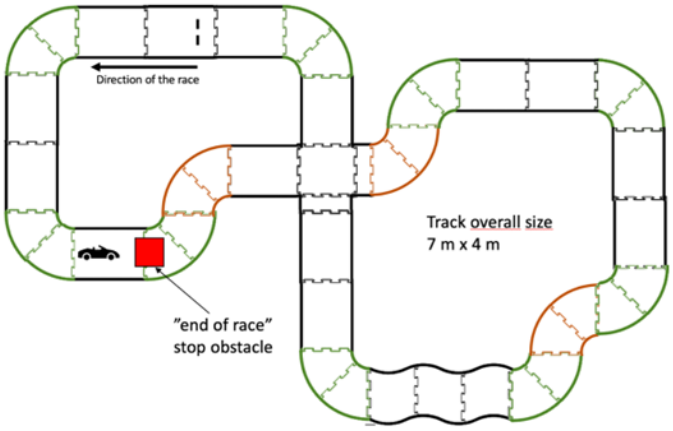


Figure 2 - Schéma Piste NXP Cup

II. Architecture matérielle et logicielle

1. Conception Électronique

L'architecture électronique du robot intègre plusieurs composants essentiels pour son fonctionnement autonome et fiable :

- **Carte de Développement NXP FRDM K22F :** Dotée d'un processeur ARM Cortex-M4, elle est compatible avec l'environnement de développement MCUXpresso.

- **Line Scan Camera parallax TSL1401-DB :** Cette caméra est utilisée pour la détection précise des lignes de la piste grâce à sa capacité à capturer une unique dimension d'image.
  Cycle de fonctionnement :
  - La broche SI démarre un nouveau cycle en envoyant une impulsion
  - Le signal CLK permet de synchroniser la lecture de chaque pixel un à un
  - Les données analogiques pour chaque pixel sont envoyées via l'ADC à chaque bord descendant de CLK
  - Après la 18ème pulsation sur CLK, la phase d'intégration commence, sa longueur détermine l'exposition des pixels de la caméra

  Il est important de stopper les pulsations sur la CLK après la lecture des 128 pixels de la caméra

- Deux **moteurs Brushless A2212/15t 930KV** pour la propulsion, contrôlés avec deux **ESC BLHELI_32**. Chaque moteur reçoit un signal PWM à une fréquence de 48 kHz, généré par les modules FTM de la carte NXP.

- **Un Servomoteur Futaba S3010 :** Utilisé pour le contrôle de la direction (braquage des roues avant), il est commandé par un signal PWM à 50 Hz. Une alimentation dédiée est nécessaire en raison de sa consommation en courant.

- **Un capteur de distance HC-SR04 :** Permet de détecter la présence d'un obstacle sur la piste. Le pin Trigger notifie le capteur qu'il doit réaliser une mesure et le pin Echo pour déterminer la distance.

- **Batterie Lipo 5000mAh, 11.1V 3C1P 55.5Wh :** Fournit l'énergie nécessaire à l'ensemble du système.

- **Régulateur de tension Power module V6.0 :** Fournit une tension régulée de 7V pour les moteurs brushless et de 5V pour l'alimentation de la carte NXP.

- **Régulateur de tension Hobbywing 3A Ubec :** Assure l'alimentation stable du servomoteur, avec un courant de sortie de 3A.

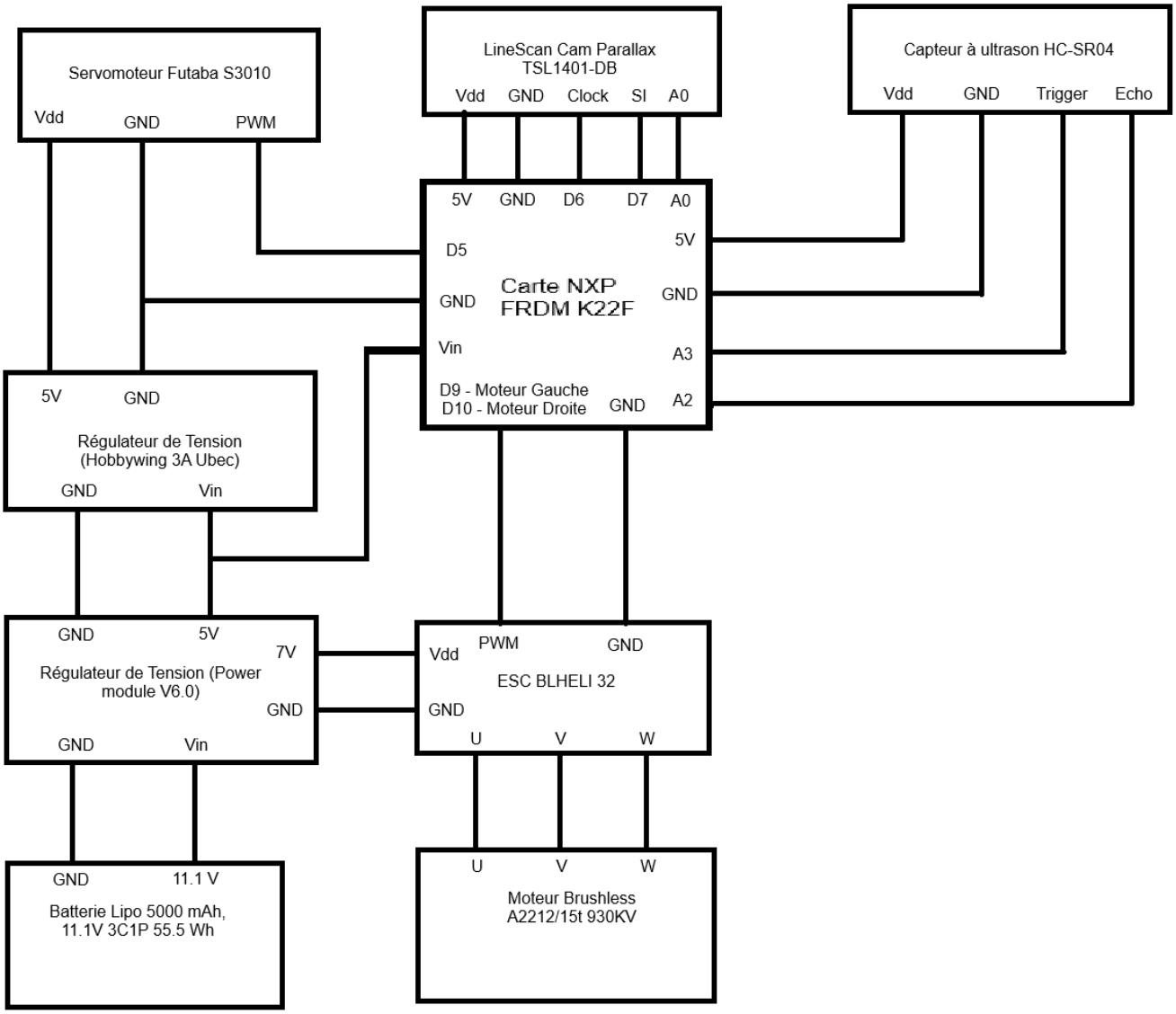


Figure 3 - Schéma électrique

2. Prise en main du logiciel

MCUXpresso IDE est l'environnement recommandé pour programmer et déboguer les cartes NXP, notamment la FRDM-K22F utilisée dans ce projet.

- Télécharger et installer l'IDE depuis le site de NXP (lien en annexe).
- Installer le SDK correspondant à la carte, nécessaire pour accéder aux outils (compilateur, drivers, exemples, etc.).
- Créer un projet via "New Project", sélectionner FRDM-K22F, puis choisir un projet vide ou basé sur un exemple.

III. Navigation

1. Fonctionnement du programme

a. Initialisation et configuration

Dans la fonction main (), tous les composants matériels sont configurés et initialisés :

- **PWM :** généré via FTM0 (moteurs gauche/droite) et FTM2 (direction) pour le contrôle des servomoteurs.
- **ADC :** utilisé pour lire les données analogiques de la caméra linéaire.
- **GPIO :** gère-les entrées/sorties numériques, notamment pour l'ultrason et le bouton.
- **SysTick :** utilisé pour les temporisations en microsecondes.

b. Perception – Suivi de ligne avec la caméra

Capteurs utilisés :

- Une caméra analogique linéaire capture 128 pixels via un signal d'horloge (clockPulse) et un signal de synchronisation (SI).
- Un capteur ultrason HC-SR04 détecte la distance entre le robot et un éventuel obstacle.

Fonctionnement :

- **getErrorFromCamera () :**
    - Capture les 128 pixels avec readADC ().
    - Utilise detectTrack () pour localiser deux lignes noires encadrant la piste.
    - Calcule le centre de la piste et en déduit l'erreur de position (différence entre le centre de l'image et celui de la piste).
- **detectTrack () :**
    - Recherche deux zones de pixels noirs séparées par une bande blanche.
    - Identifie les bords gauche et droit de la piste pour déterminer la trajectoire à suivre.

c. Contrôle – Ajustement de la direction (PID)

La fonction turn () utilise un **contrôleur PID** pour ajuster dynamiquement l'angle de direction :

- Le PID calcule un angle correcteur basé sur l'erreur actuelle, son intégrale, et sa dérivée.
- Un mécanisme anti Wind-up limite l'accumulation de l'intégrale.
- Le résultat est utilisé pour modifier le rapport cyclique (duty cycle) du servomoteur de direction.

Logique de manœuvre :

- Variables turnLeft, turnRight et middle définissent la situation (virage ou ligne droite).
- updatedDutycycleTurn ajuste la direction ; updatedDutycycleLeft et updatedDutycycleRight modifient la vitesse de chaque moteur.
    - Exemple : en virage à gauche (turnLeft), la roue gauche ralentit par rapport à la droite.
    - En ligne droite (middle), les deux moteurs tournent à vitesse égale.

d. Commande moteur – Vitesse

- UpdatePwmDutycycleSpeed () applique les PWM aux deux moteurs brushless selon les valeurs calculées.
- UpdatePwmDutycycleTurn () applique le PWM au servomoteur de direction.

e. Évitement d'obstacles – Capteur ultrason

Le capteur HC-SR04 permet de mesurer la distance à un obstacle :

- sendTrigger () : envoie une impulsion pour initier la mesure.
- ECHO_IRQ_HANDLER () : calcule la distance à partir du temps de réponse (temps aller-retour de l'onde ultrasonore).

Comportement :

- Si la distance est inférieure à 20 cm, la fonction motorBreak () est appelée :
    - Le robot s'arrête.
    - Il attend soit que l'obstacle disparaisse, soit qu'un bouton soit pressé pour redémarrer.

f. Boucle Principale

Dans la boucle while (1) de main () :

- Lecture périodique du capteur ultrason.
- Lecture des pixels de la caméra et calcul de l'erreur de trajectoire.
- Ajustement de la direction avec turn ().
- Mise à jour de la vitesse selon la trajectoire détectée.

2. Contrôle des moteurs

a. Moteurs Brushless

Pour piloter les moteurs brushless, nous utilisons des signaux PWM générés via les modules FTM (FlexTimerModule), configurés à l'aide des fonctions fournies dans le SDK NXP.

- **Fréquence du PWM :**
  Selon la documentation technique des ESC BLHELI_32, la fréquence maximale supportée est de 48 kHz. En l'absence de contraintes spécifiques, nous avons choisi d'utiliser cette fréquence pour optimiser la réactivité du moteur.
- **Plage de fonctionnement (duty cycle) :**
  Une phase de test a permis de déterminer que les moteurs fonctionnent efficacement dans une plage comprise entre 30 % et 65 % de rapport cyclique.
  - 30 % à 46 % : marche arrière
  - 47 % : arrêt
  - 48 % à 65 % : marche avant

Les ESC ont été configurés en conséquence pour autoriser à la fois le mouvement avant et arrière.

b. Servomoteur

Le servomoteur Futaba S3010, utilisé pour la direction, est également contrôlé par un signal PWM, selon une approche similaire.

- **Fréquence du PWM :**
  D'après la datasheet, le signal de commande doit avoir une fréquence de 50 Hz. Un canal FTM dédié a donc été initialisé avec cette fréquence.

- **Duty cycle :**
  La plage utile de rapport cyclique a été déterminée expérimentalement. Comme le servomoteur a une limite mécanique d'angle, il est essentiel de ne pas dépasser cette plage pour éviter de forcer ou d'endommager le mécanisme.

IV. Algorithme

1. Améliorations apportées

1. detectTrack () :

L'ancienne version de la fonction souffrait de détections erronées dans les cas où le sol était sombre : la bordure gauche était souvent placée à la position 0, faussant l'analyse. De plus, lorsqu'une seule bordure était détectée, sa position pouvait être mal interprétée (ex. bordure gauche placée à droite de l'image = mauvais virage).

Améliorations apportées :

- La détection commence désormais par repérer la piste blanche, puis les bordures noires.
- En cas de détection unique, la position du bord non blanc de la caméra permet de déduire s'il s'agit de la bordure gauche ou droite.
- Ces changements rendent la détection plus robuste, notamment dans les environnements sombres.

2. getErrorFromCamera () :

L'ancienne version attribuait arbitrairement une erreur de +20 lorsque la bordure gauche était mal détectée (souvent placée à 0), pour forcer un virage à gauche.

Améliorations apportées :

- Suppression des valeurs arbitraires.
- Calcul de l'erreur basé uniquement sur les vraies positions détectées par detectTrack ().
- Résultat : une meilleure précision dans le suivi de trajectoire.

3. Turn () :

L'ancienne fonction utilisait trois valeurs fixes pour contrôler la direction du servomoteur :

- Gauche : 8
- Droite : 13
- Tout droit : 11

Ce système donnait un comportement rigide et peu fluide.

Améliorations apportées :

- Intégration d'un contrôleur PID pour calculer dynamiquement l'angle de correction.
- Mise en place d'un anti Wind-up pour stabiliser l'intégrale.
- Mapping linéaire de la consigne PWM → direction plus précise et fluide.

2. Axes d’amélioration

Bien que le robot soit fonctionnel, certaines limitations restent à corriger :

- **Sensibilité à la lumière :**
  Le capteur linescan est très sensible aux variations d’éclairage. L’utilisation de seuils fixes (noir/blanc) rend le système instable dans des conditions lumineuses non idéales. Une calibration automatique ou dynamique serait bénéfique.

- **Réglage fin du PID :**
  Les constantes du PID sont encore basiques et nécessitent un tuning précis pour optimiser le comportement du robot.

- **Dépendance à un seul capteur :**
  Le robot utilise uniquement la caméra linescan. Une perte de contraste ou une mauvaise calibration peut empêcher complètement la navigation. L’ajout d’autres capteurs (IR, infrarouge, LIDAR...) améliorerait la robustesse.

- **Détection de la ligne d’arrivée :**
  Actuellement gérée par la linescan, ce qui peut interférer avec la navigation : le robot interprète parfois la ligne d’arrivée comme un virage. Un module indépendant de détection (capteur infrarouge ou caméra secondaire) permettrait de mieux gérer cet événement.

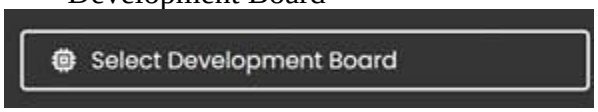
V. Annexe

Liens utiles :

- Datasheet esc BLHeli_32 : https://img.banggood.com/file/products/20171201012630BLHeli_32%20manual%20ARM%20Rev32.x.pdf
- datasheet Brushless Moteur – A2212/15T – 930KV : https://bc-robotics.com/shop/brushless-motor-a221215t-930kv/
- datasheet Servomoteur https://futabausa.com/wphttps://futabausa.com/wp-content/uploads/2018/09/S3010.pdfcontent/uploads/2018/09/S3010.pdf
- GitHub du groupe de Guillaume RIO et Oumnia BHIKIR : https://github.com/RIOguillaume/NXPCup

Téléchargement :

- MCUxpresso https://www.nxp.com/design/design-center/software/developmenthttps://www.nxp.com/design/design-center/software/development-software/mcuxpresso-software-and-tools-/mcuxpresso-integrated-development-environment-ide:MCUXpresso-IDEsoftware/mcuxpres so-software-and-tools-/mcuxpresso-integrated-developmenthttps://www.nxp.com/design/design-center/software/development-software/mcuxpresso-software-and-tools-/mcuxpresso-integrated-development-environment-ide:MCUXpresso-IDEenvironment-ide:MCUXpresso-IDE
- SDK pour la carte NXP FRDM K22F
  - Se rendre sur cette page https://mcuxpresso.nxp.com/en ● Cliquer sur le bouton select Development Board

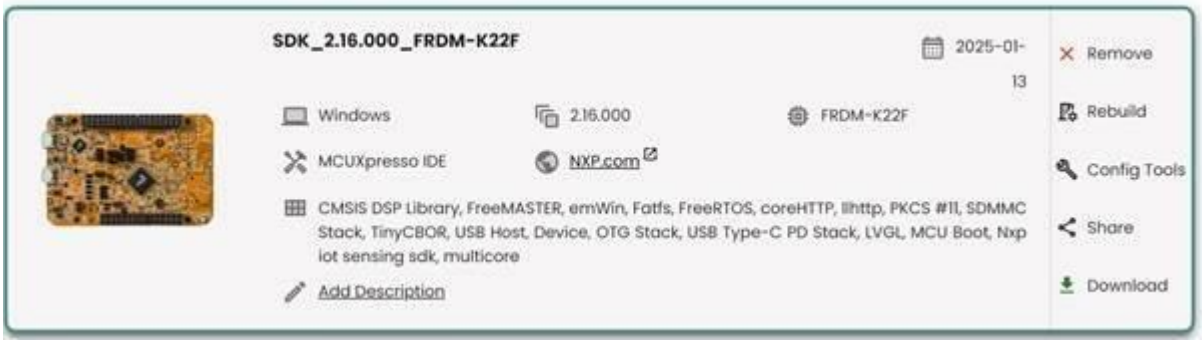


  - Se connecter à NXP (créer un compte si vous n'en possédez pas)
  - Dans la barre de recherche entré FRDM K22F

  - Cliquer sur la carte dans l'onglet Board

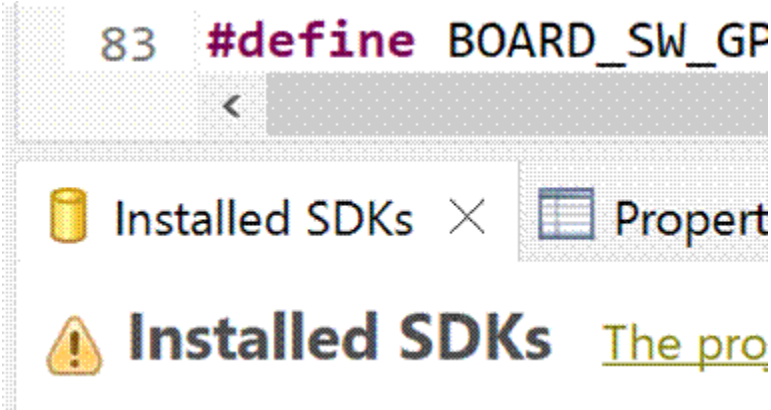
Cette page devrait se trouver en dessous (il faut scroller vers le bas)

- Vous pouvez cliquer sur build SDK,
- Choisissez vos paramètres en cochant les différentes options et en définissant votre OS (nous avons coché toutes les cases pour ne pas nous prendre la tête).
- Vous obtenez une page avec différentes archives, vous pouvez cliquer sur le bouton download en bas à droite



Pour importer le SDK dans MCUxpresso

- En bas choisir l'onglet Installed SDKs



- Puis cliquer sur la petite flèche à droite pour importer votre archive ou cliquer déposer votre archive directement dans l'onglet installed SDK

- Vous devriez obtenir quelque chose comme ça :

Description des pins de la carte :