Rapport final de projet

Année scolaire 2024-2025

NXPCup

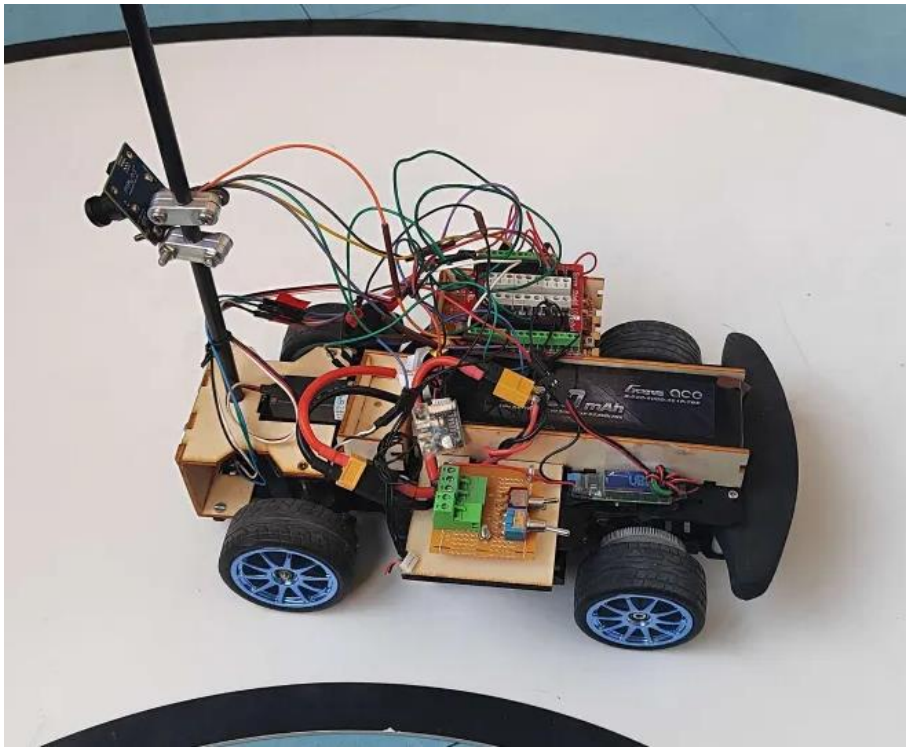


Figure 1 - Image du Robot

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. NXP Cup

La NXP Cup est une compétition étudiante internationale organisée par NXP Semiconductors. Elle vise à encourager les étudiants à développer des compétences en technologie, ingénierie et programmation en concevant et programmant une voiture autonome. Les participants doivent construire un véhicule capable de naviguer sur un circuit inconnu, délimité par des lignes noires, de manière totalement autonome.

C'est une course de vitesse, où la voiture doit compléter le circuit le plus rapidement possible. Et une épreuve de précision, où la voiture doit s'arrêter à moins de 10 cm d'un obstacle sur la piste après un tour rapide.

Les voitures doivent être construites avec des composants NXP et programmées exclusivement avec l'environnement MCUXpresso.

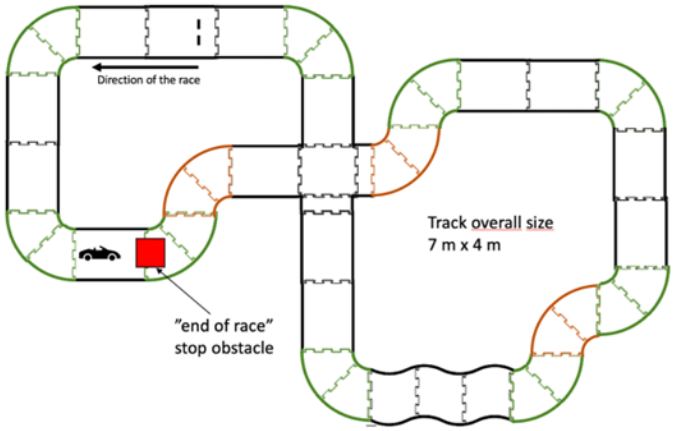


Figure 2 - Schéma Piste NXPCup

Concernant les règles de la course, la voiture doit garder au moins 2 roues sur la piste tout au long de son trajet.

Après avoir atteint la ligne d'arrivée, la voiture doit visiblement ralentir pour ensuite s'arrêter à 10 cm de l'obstacle.

II. Matériel et logiciel

Pour mener à bien le projet, cette section détaillera les outils nécessaires pour reproduire, comprendre et étendre notre travail.

a. Liste du matériel

- Carte de Développement **NXP FRDM K22F** avec processeur ARM Cortex M4 et compatible avec l'environnement de développement MCUXpresso.
- **Line Scan Camera parallax TSL1401-DB** (pour une détection précise des lignes grâce à sa capacité à capturer une dimension unique des données)
- Deux moteurs **Brushless a2212/15t 930kv** pour la propulsion, contrôlés avec deux **ESC blheli_32**.
- Un **Servomoteur futaba s3010**, pour gérer la rotation de la voiture.
- Un **capteur de distance HC-SR04**, la course se déroule sur deux temps, lors du deuxième tour de piste, les voitures doivent s'arrêter à une distance prédéfinie de 10 cm devant un obstacle, placé sur la piste par les juges.
- Batterie et régulateurs de tension : nous possédons une batterie **Lipo 5000mAh, 11.1V 3C1P 55.5Wh**. Nous utilisons un régulateur de tension (**Power module V6.0**) afin de générer une tension de 7V pour les moteurs brushless et une tension de 5V pour alimenter la carte. Un autre régulateur de tension (**hobbywing 3a ubec**) uniquement pour alimenter le Servomoteur qui nécessite 3A.
- Structure du véhicule, châssis fournis par NXP avec guide pour montage.
- Câbles et connecteurs.

b. Prise en main du logiciel

MCUXpresso est l'environnement de développement intégré recommandé pour programmer et déboguer les cartes NXP, y compris la FRDM utilisée dans ce projet. Cette section explique comment prendre en main cet outil.

- Installation MCUXpresso : sur le site officiel de NXP, télécharger la dernière version de MCUXpresso IDE, en suivant les instructions d'installation adapté au système d'exploitation utilisé. (Lien en annexe)
- Importance du SDK dans MCUXPresso : Un SDK est un kit de développement logiciel. Il contient un ensemble d'outils de création spécifiques à notre carte. Durant la programmation vous aurez besoin de composants tels que des débogueurs, des compilateurs, des drivers et des bibliothèques contenant des exemples, tout ceci est contenu dans le SDK.
- Télécharger le SDK pour la FRDM-K22F : Il vous est donc impératif de télécharger le SDK correspondant à votre carte afin de pouvoir créer et exécuter un programme sur votre carte. (Explication de la procédure et lien en annexe)
- Création projet : Si vous souhaitez recommencer un projet vous pouvez aller dans le menu principal, cliquez sur New Projet, sélectionnez la carte FRDM K22F et configurez un projet vide ou existant.

III. Circuit électrique

a. Connexion de la Line Scan Camera via ADC

La parallax TSL1401-DB est une caméra linéaire capable de capturer des informations en une seule dimension. Elle utilise une broche analogique (A0) pour transmettre les données et des broches numériques pour les signaux de synchronisation.

Description des connexions :

- GND : connecté à la masse de la carte -
- Vdd : alimentation de la caméra (3.3V Arduino, 5V NXP)
- A0 : sortie analogique connectée à l'entrée ADC de la carte
- SI : entrée numérique pour la synchronisation, GPIO configuré en sortie
- CLK : entrée numérique pour l'horloge, GPIO configuré en sortie

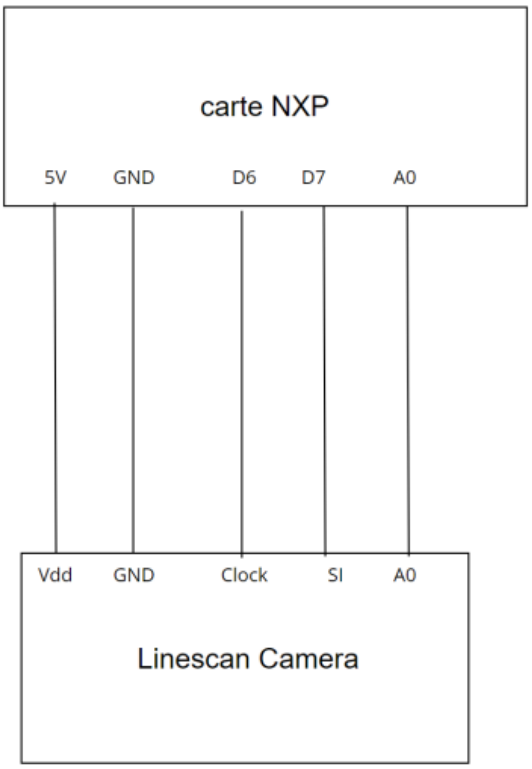


Figure 3 - Configuration pin Linescan

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

b. Connexion du capteur à ultrason hc-sr04

Le capteur à ultrason est utilisé pour détecter s'il y a un obstacle sur la piste. Pour ce faire on utilise deux pins, le pin Trigger afin de notifier le capteur qu'il doit réaliser une mesure et le pin Echo pour déterminer la distance.

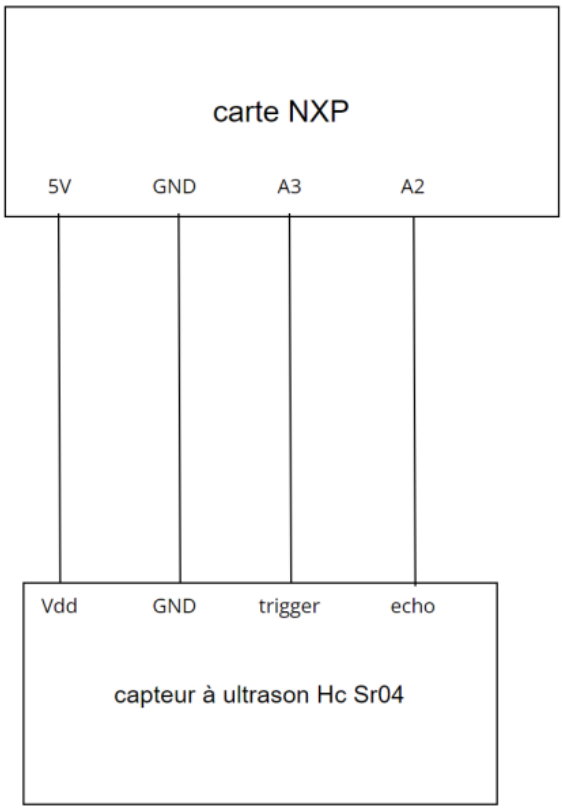


Figure 4 - Configuration pin HC-SR04

Cycle de fonctionnement :

Lorsque le pin trigger est à l'état haut, le capteur envoie 8 impulsions à 40Khz et place le pin Echo à l'état haut. Lors de la réception d'une des impulsions, le capteur passe l'état du pin Echo à bas. Ainsi en mesurant le temps que le pin Echo est resté à haut, on peut déterminer la distance.

c. Connexion du servomoteur

Le servomoteur contrôle la direction en ajustant les roues avant. Il est également commandé par un signal PWM (Fréquence : 50Hz). Pour ce faire, on utilise 1 pins, le pin PWM pour contrôler la vitesse et la direction des moteurs.

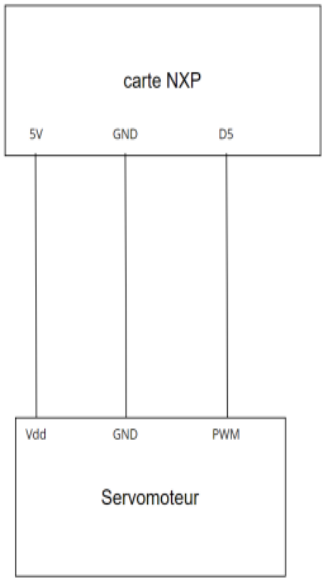


Figure 5 - Configuration pin servomoteur

d. Connexion des ESC

Les moteurs brushless contrôlent les roues arrière et fournissent la propulsion nécessaire.
Leur vitesse est régulée par des signaux PWM envoyés via les ESC.
Chaque moteur est contrôlé par un signal PWM (Fréquence : 48 kHz) généré par les FTM de la carte NXP, les broches utilisées sont les suivantes :

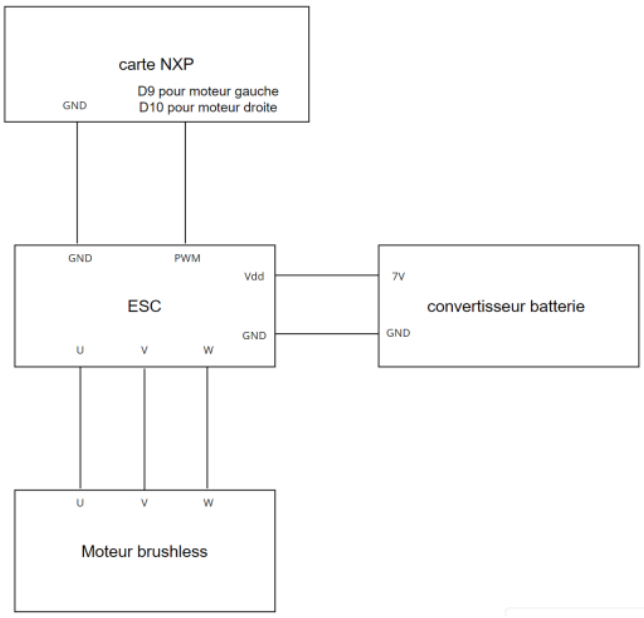


Figure 6 - Configuration pin moteur brushless

IV. Navigation

a. Traitement d'image

Le traitement d'image et la détection de la piste sont au cœur de ce projet, ils permettent au véhicule de naviguer sur le circuit. Cette section détaille les étapes pour analyser les données de la linescan camera, détecter les bords de la piste.

- Structure des Données de la Linescan Camera :
La linescan camera fournit une séquence de 128 pixels, chaque pixel représentant une intensité lumineuse capturée. Les valeurs analogiques varient généralement de 0 (noir) à 4095 (blanc). Cette information permet de repérer les zones blanches (piste) et noires (bordures) sur le circuit.

- Détection des Bordures :
Les valeurs analogiques sont lues via l'ADC et stockées dans un tableau, un filtre exponentiel est appliqué pour réduire le bruit et lisser les données. On parcourt le tableau pour récupérer les transistions noir-blanc (bordure gauche) et blanc-noir (bordure droite). Les indices de ces transitions sont enregistrés pour identifier les bords de la piste.

- Calcule de la trajectoire :
Le centre de la piste est calculé comme moyenne des indices des bordures détectées : centre = (leftEdge + rightEdge) / 2
L'erreur de position est ensuite définie comme la différence entre le centre de la caméra (64 pour 128 pixels) et le centre de la piste détectée.

- Direction à suivre :
- L'erreur est gérée par le contrôleur, celui-ci fait tourner le servomoteur de manière à annuler l'erreur.
- Gestion des cas particuliers :
    - Une seule bordure détectée : Si seulement le bord gauche est détecté, la bordure droite est placée sur le bord droit du tableau (la bordure droite aura une valeur de 128). Pareillement pour le bord droit (qui aura une valeur de 0).
    - Aucune bordure détectée : Ce cas est le plus complexe, en effet il peut indiquer plusieurs choses : une sortie de piste, un virage pris trop large en raison d'une grande vitesse et d'un PID mal réglé ou d'une détection trop lente des bordures ou encore une intersection, dans tous les cas le robot doit continuer à suivre la consigne précédente. On applique donc l'erreur précédente tout en ralentissant légèrement le robot.
- Si le signal est instable ou bruité : on vérifie la calibration de la caméra et les branchements avec un oscilloscope. Un filtre a été ajouté dans le but de lisser le signal.

- Implémentation de la caméra :
Le code pour la caméra comprend plusieurs fonctions essentielles :
clockPulse(), delay(), initADC(), readADC(), readFilteredADC(), detectTrack(), getErrorFromCamera() et adjustDirection().

- La fonction clockPulse() envoie une impulsion d'horloge à la caméra. Elle règle la broche d'horloge à l'état haut pendant 1 microseconde, puis à l'état bas pour la même durée. Cette impulsion est nécessaire pour synchroniser la lecture des pixels avec le capteur de la caméra.

- La fonction delay() introduit un délai en exécutant une instruction NOP (No Operation) un certain nombre de fois. Cela permet de contrôler le timing des opérations cruciales pour la synchronisation avec la caméra.

- La fonction initADC() initialise le convertisseur analogique-numérique (ADC) sur la carte. Elle configure l'ADC avec les paramètres par défaut, exécute une calibration automatique et prépare la structure de configuration du canal pour la lecture des valeurs analogiques.

- La fonction readADC() lit une valeur analogique à partir du capteur de la caméra via l'ADC. Elle configure le canal de l'ADC, attend que la conversion soit terminée, puis retourne la valeur convertie.

- La fonction detectTrack() analyse les valeurs des pixels pour détecter les bords de la piste. Elle recherche deux séquences de pixels noirs séparées par une distance minimale, représentant les bords gauche et droit de la piste. Si elle ne trouve qu'une seule ligne, elle détermine si le robot est trop à gauche ou à droite.

- La fonction getErrorFromCamera() trouve le centre de la piste et appelle la fonction adjustDirection() en fonction de celui-ci.

- La fonction adjustDirection() ajuste la direction du robot en fonction de la position détectée de la piste. Elle calcule la déviation par rapport au centre de l'image et imprime la direction à suivre.

En résumé, notre implémentation initialise et configure l'ADC, lit les valeurs des pixels de la caméra, détecte les bords de la piste et ajuste la direction du robot en conséquence. Ces fonctions travaillent ensemble pour permettre au robot de suivre la piste de manière autonome en utilisant les données de la caméra.

Résultats observés :

La linescan caméra ne renvoie pas d’image visuelle, elle capture des images lignes par lignes plutôt que par matrice de pixels comme les caméras traditionnelles, chaque pixel de cette ligne capture la lumière en petite portion, et l’horloge contrôle la fréquence à laquelle le pixel est lu. Ainsi il est impossible de visualiser ce que voit la caméra. Nous avons donc décidé d’afficher la valeur de l’intensité allant de 0 pour le noir à 4095 valeurs pour le blanc pur. Ces valeurs lues séquentiellement à chaque cycle d’horloge sont ensuite traitées pour former une image complète ligne par ligne.

b. Contrôles moteurs

Moteurs brushless :

Afin de faire fonctionner nos moteurs brushless, il nous suffit de générer un signal PWM. Pour ce faire nous utilisons des channels FTM (FlexTimerModule) que nous configurons à l'aide des fonctions mise à disposition dans le SDK. Afin de choisir la fréquence que l'on va utiliser pour notre PWM nous sommes allés dans la documentation de l'ESC et la fréquence maximale qu'il supporte est de 48kHz. Nous ne possédons pas de contrainte spécifique, nous avons donc décidé de les contrôler à cette fréquence.

Ensuite, il fallait déterminer la range de fonctionnement de nos moteurs (le dutycycle du PWM). De manière expérimental nous avons déterminé que la range de fonctionnement était entre 30% et 65% du dutycycle.

De plus, nous avons configuré nos esc afin que nos moteurs puissent avancer et reculer. Ainsi si le dutycylce est compris entre 30 et 46 nos moteurs recule, entre 48 et 65 nos moteurs avance et nos moteurs sont arrêtés pour 47.

Servo moteur :

Pour le servomoteur nous avons suivi la même méthodologie que pour les moteurs brushless. Afin de déterminer la fréquence de fonctionnement nous sommes allés voir la datasheet du servomoteur et nous avons trouvé que pour le contrôler il faut un signal PWM à 50Hz. Nous avons donc initialisé un autre channel cette fréquence.

Enfin nous avons déterminé le dutycycle expérimentalement car l'angle de rotation est limité mécaniquement il ne faut donc pas envoyer un signal qui pourrait faire forcer le servomoteur et le casser.

V. Algorithme

a. Améliorations apportées

Notre groupe a pu apporter plusieurs améliorations au projet :

- Amélioration de la logique de la fonction detectTrack() qui auparavant détectait une intersection sur les premiers pixels de la caméra si le sol était noir.
- Amélioration de la logique de la fonction getErrorFromCamera() qui pour compenser l'erreur précédente tournait à droite uniquement dans le cas où la bordure gauche n'était pas détectée.
- Amélioration de la logique de la fonction turn() qui utilisait seulement 3 positions pour le servo moteur. Quand le robot devait tourner les roues prenaient un angle de 45° et quand le robot devait aller tout droit elles prenaient un angle de 0°. On a donc remplacé cette logique par un contrôleur PID qui corrige l'erreur obtenue avec getErrorFromCamera().

b. A améliorer

Il reste cependant des choses à améliorer :

- Sensibilité trop élevée à la lumière, la caméra linescan est par défaut très sensible aux conditions d'éclairage, de plus le programme utilise pour l'instant des seuils fixes pour la détection du noir et du blanc ce qui fait que si la pièce n'est pas très bien éclairée, le robot ne détectera pas les bords de la piste.

- Amélioration des constantes du PID

- Utilisation d'un seul capteur, le robot n'utilise que la linescan caméra pour suivre la piste, cela le rend extrêmement dépendant, si la calibration ou les conditions d'éclairage ne sont pas suffisantes pour la linescan caméra le robot ne pourra pas opérer.

- Détection de la ligne de fin, pour l'instant la linecam s'occupe de la détection de la ligne de fin cependant, cela peut interférer avec le programme de navigation et, dans certains cas, le robot tournera au lieu de simplement ralentir.

VI. 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 Servo Moteur https://futabausa.com/wp-content/uploads/2018/09/S3010.pdf

Téléchargement:

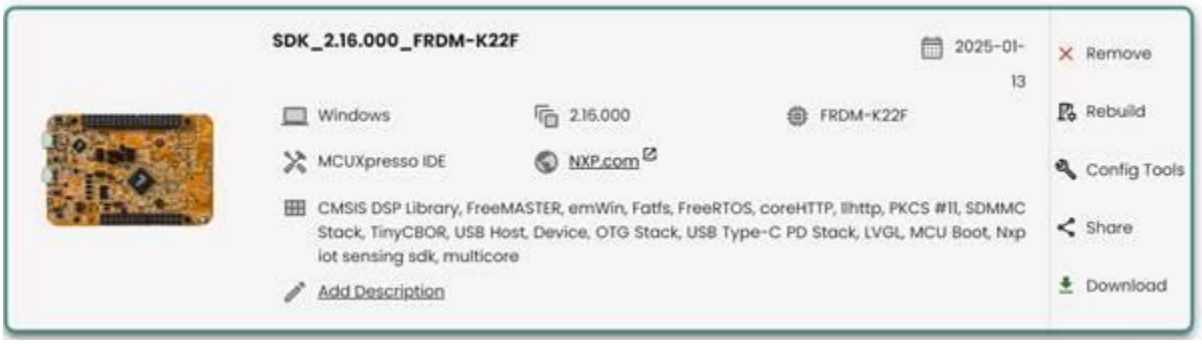
- MCUxpresso https://www.nxp.com/design/design-center/software/development-software/mcuxpres so-software-and-tools-/mcuxpresso-integrated-development-environment-ide:MCUXp resso-IDE
- SDK pour la carte NXP FRDM K22F
  - Se rendre sur cette page https://mcuxpresso.nxp.com/en
  - Cliquer sur le bouton 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

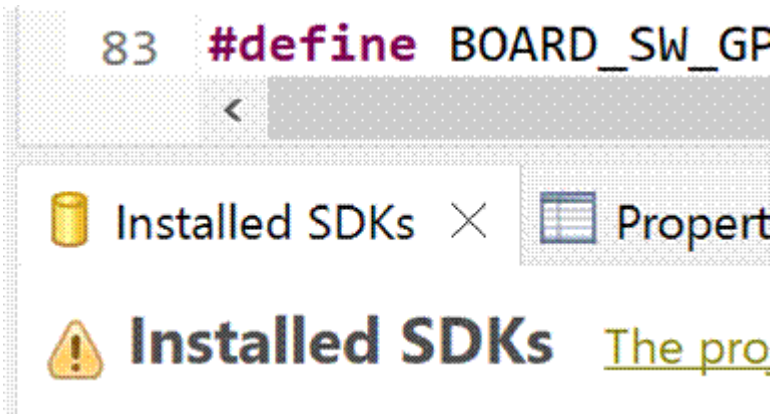
Cette page devrait se trouver en dessous (il faut scroller vers le bas)

- 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

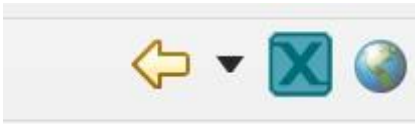


Pour importer le SDK dans MCUxpresso

- en bas choisir l'onglet Installed SDKs

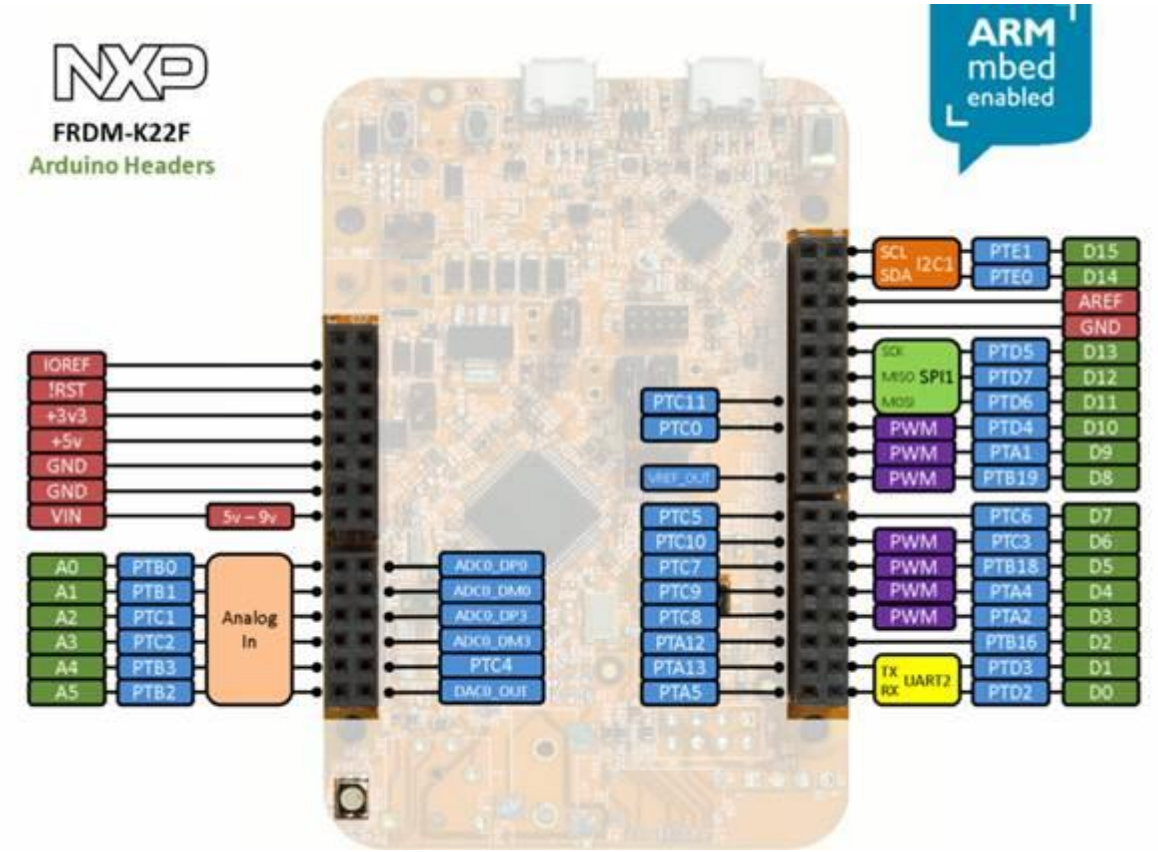


- puis cliquer sur la petite flèche à droite pour importer votre archive ou cliquer déposer votre archive directement dans l'onglet installed SDK



- Vous devriez obtenir quelque chose comme ça :

Description des pins de la carte :


NXP
FRDM-K22F
Arduino Headers
ARM
mbed
enabled
IOREF
!RST
+3v3
+5v
GND
GND
VIN
5v – 9v
A0
A1
A2
A3
A4
A5
PTB0
PTB1
PTC1
PTC2
PTB3
PTB2
Analog
In
ADC0_DP0
ADC0_DM0
ADC0_DP3
ADC0_DM3
PTC4
DAC0_OUT
PTC11
PTC0
VREF_OUT
PTC5
PTC10
PTC7
PTC9
PTC8
PTA12
PTA13
PTA5
SCL
SDA
I2C1
PTE1
PTE0
D15
D14
AREF
GND
SCK
MISO
MOSI
SPI1
PTD5
PTD7
PTD6
PTD4
PTA1
PTB19
D13
D12
D11
D10
D9
D8
PWM
PWM
PWM
PWM
PWM
PWM
PWM
PWM
PTC6
PTC3
PTB18
PTA4
PTA2
PTB16
PTD3
PTD2
D7
D6
D5
D4
D3
D2
D1
D0
TX
RX
UART2