# Rapport de conception
# Carte mère Robot Principal

Les passages en rouge concernent les aspects qui sont encore incomplets.

**Introduction :**

Le but de ce document est de récapituler l'avancement de ce projet, ainsi que de justifier mes choix de conception. Il s'agira a posteriori de revenir sur ces choix afin d'en tirer des conclusions utiles pour la suite des projets. Ce document permettra avant tout de garder une trace des réflexions et questions qui se sont posées lors de la conception.

Pour replacer le contexte, il s'agit de concevoir la carte mère du robot principal de RTS pour la Coupe de Robotique 2019, en Belgique puis en France. N'ayant quasi aucun des composants qui seront utilisés sur la carte, ni beaucoup d'expérience dans les technologies utilisées, il a fallu avancer à tâtons. Le projet a commencé en janvier 2019, et rien du robot n'avait été conçu, le cahier des charges est donc très peu développé, ce qui justifie notre choix de garder un maximum de flexibilité en multipliant les ajouts de composants et de connexions afin de garder un maximum de degré de liberté une fois le circuit entre les mains.

**Teensy 3.6 :**

Il s'agit de la carte de développement choisie pour piloter tout le robot. Pourvue d'un processeur ARM 180 MHz (Contre 16 MHz pour l'ATmega 2560 d'une Arduino Mega), elle s'utilise exactement comme une Arduino standard. Toutes les entrées digitales peuvent servir d'entrée d'interruption. J'ai ajouté deux rangées de connecteurs reliées aux pins latérales de la carte, pour remplir l'objectif de flexibilité.

**Ressources :**

Specs :https://www.pjrc.com/teensy/techspecs.html
Pinout : https://www.pjrc.com/teensy/pinout.html
Datasheet microcontrôleur: https://www.pjrc.com/teensy/K66P144M180SF5RMV2.pdf

**Alimentation :**

La *Teensy* n'acceptant qu'une alimentation de 3.3V, la régulation se fait par un régulateur tout à fait standard. Afin d'assurer la sécurité du montage, des condensateurs au Tantalum doivent être ajoutés. Ajouter modèle régulateur choisi.

**Raspberry Pi :**

Non apparente sur les schémas, la Raspberry Pi se chargera de gérer la caméra et le Lidar du robot. Elle doit donc échanger avec la Teensy. La communication I2C entre les deux est théoriquement possible, mais il est complexe de configurer la Raspberry Pi en esclave I2C (cela dépend largement de la version de la R.Pi, et la documentation est pas toujours facile à trouver). Nous avons donc opté pour une communication Série (alias UART) entre les deux. Puisque les niveaux logiques des deux appareils sont égaux à 3.3V, aucune adaptation de tension logique n'est à effectuer, il suffit d'assurer les branchements RX/TX.
Des documents sur le bus Série seront rédigés pour les formations.

**Bus I2C :**

Certains composants, notamment les modules encodeurs magnétiques *(AS5601)* sont censés communiquer avec la Teensy par bus I2C. Puisqu'ils ont tous la même adresse I2C, et qu'on en utilise deux, il est nécessaire d'utiliser un multiplexeur I2C pour permettre de les utiliser. On utilisera le *TCA9548A*, multiplexeur 1 vers 8. Celui-ci contient déjà des résistances de pull-up égales à 10kR du côté du master (donc *Teensy*), mais je garde la possibilité de les améliorer en posant des résistances externes au multiplexeur. En revanche, il n'en est rien du côté des esclaves, c'est pour cela qu'elles sont également ajoutées.
Par exemple, les ***SDA*** et ***SCL*** des encodeurs *AS5601*sont en drain ouvert, et nécessitent donc l'ajout de résistances de pull-up. Les premiers tests concluants ont été réalisés avec des résistances de valeur 4.7kR. Un accès au canal I2C hors multiplexeur est également disponible.
En sortie du multiplexeur, pour s'assurer d'une bonne transmission logique, le **GND** et **Vcc** (3.3V) sont ajoutés pour les tensions de référence.
« The device offers an active-low [/inverted] **RESET** » : il faut ajouter une résistance de pull-up pour empêcher le **RESET** en fonctionnement nominal.
***[cf p.15, 9.5.4 & p.18, 10.2 : Typical Application]***
*Borne connectée à la Teensy (A13, pin 32).*

Des documents sur le bus I2C seront rédigés pour les formations.

*Lien Multiplexeur (Breakout) :*
*https://learn.adafruit.com/adafruit-tca9548a-1-to-8-i2c-multiplexer-breakout/overview*
*Datasheet TCA9548A :*
*https://cdn-shop.adafruit.com/datasheets/tca9548a.pdf*

**Drivers Moteurs Pas à Pas :**

Pins de mode en **pull-up** 100kΩ : ne pas brancher → niveau logique 1.
Le DRV8825 est un microstepping driver, il permet donc de réduire la taille d'un pas de moteur pas à pas. Cette fonctionnalité est accessible en touchant aux broches de mode, **M0**, **M1** et **M2**, voir le tableau de correspondance sur le lien en-dessous. Ces bornes sont en pull-down, donc en niveau bas si non branchées.
« The resolution (step size) selector inputs (MODE0, MODE1, and MODE2) enable selection from the six step resolutions according to the table below. All three selector inputs have internal 100kΩ pull-down resistors, so leaving these three microstep selection pins disconnected results in full-step mode. » (D'après lien en fin de paragraphe). Confirmé par le datasheet du DRV8825 [https://www.pololu.com/file/0J590/drv8825.pdf, p.3, Pin Functions tab]
Il s'agit donc de se garder la possibilité de connecter les bornes à **VCC** via des jumpers, d'où pour chaque drivers les connecteurs 3x2.
Les bornes **ENABLE** sont inversées et reliées à des résistances de **pull-down**. Le driver est donc actif lorsque **ENABLE** est à niveau bas, ce qui est toujours le cas avec la résistance de **pull-down**. On va utiliser des cavaliers de la même manière que pour les modes, pour pouvoir éventuellement relier ces **ENABLE** à **Vcc** et désactiver les drivers.
(cf « **Disabling pins** » sur schéma)
**Sleep** et **Reset** connectés à Vcc (+3.3V), car comme il s'agit de bornes inversées, on les désactive ainsi.
24 à 27 en **DIR**, puis 28 à 31 en **STP.**

**Fault** connectés à des connecteurs JST, pour récupérer le signal pour des LEDs extérieures à la carte mère. Puisque le niveau de ces sorties est inversé, ie Haut quand il n'y a pas de probème, et bas quand il y en a, les LEDs seront allumées en état nominal.
*Lien DRV8825 :*
*https://www.pololu.com/product/2133*

**Drivers Moteurs DC :**

Le drivers choisis sont les *MD10C* de Cytron Technologies.
Capables de délivrer des courants très élevés, il suffit de deux connexions à la *Teensy* pour les utiliser (**DIR** pour la direction, et **PWM** pour la vitesse).
Les branchements pour quatre drivers sont présents sur la carte, deux sont dédiées au roues de la base roulante, les deux autres laissées si le besoin se fait finalement sentir.
*Lien des drivers :*
*https://www.robotshop.com/eu/fr/controleur-un-moteur-13a-5-a-30v-cytron.html*

**Choix des pins de connexion *Teensy*/Composants :**

Il est tentant de se dire qu'il importe peu de brancher la plupart des composants à des endroits bien précis de la carte, tant qu'ils assurent la bonne fonction. D'autant plus que les E/S sont nombreuses sur la *Teensy*. Mais à partir du moment où beaucoup de fonctions communes à certaines E/S sont utilisées, le choix est beaucoup moins simple. S'assurer que les PWM utilisés ne sont pas également les connexions du bus Série ou I2C, par exemple, est un détail évident mais à ne surtout pas oublier. Il convient de prioriser les E/S les plus rares, comme celles des bus I2C, au détriment des plus nombreuses, comme les PWM.
*La quantité de pins analogiques est suffisamment grande pour se permettre d'en utiliser à d'autres fins, à savoir 22 sur la face supérieures, facilement accessibles. C'est pour cela que les broches d'adresse du multiplexeur, ainsi que les broches de direction des drivers DC sont branchés sur les bornes A0 à A6.*

**Affichage :**

À l'instar d'une Arduino, placer des LEDs de debug sur la carte mère.
+LED d'état d'alimentation
*A faire*

**Accès E/S inférieures de la *Teensy* :**

*Les infos sont recueillies, reste à rédiger*

**Dynamixel :**

~~Ajout d'un canal Série entre la série et le réseau de Dynamixel.~~
La Raspberry Pi s'occupera des Dynamixels.

*Les prochaines parties sont peu importantes, car reflètent simplement la flexibilité visée pour la carte :*

**SPI :**

Deux blocs de connexion pouvant assurer l'utilisation de bus SPI sont mis à disposition.