Rapport final de projet de 6e session

Ajouter une page de présentation !!!

# Projet

## Explication du projet

Le but du projet était de construire deux véhicules téléguidés ayant la capacité d’être contrôlés de façon manuelle, autonome ou semi-autonome. La télécommande, aussi construite et désigné sur mesure, devait permettre de choisir le mode de contrôle de chaque véhicule, tout en s’occupant du contrôle manuel dans le cas où le mode manuel était sélectionné.

Chaque véhicule devait être capable d’envoyer un flot d’image prise par leur caméra respective via une communication Wi-Fi. La télécommande devait aussi avoir la capacité de contrôler l’orientation des caméras à l’aide de servo moteurs placés sur le dessus des véhicules.

Chaque véhicule devait tirer sa source d’énergie principale de batteries au lithium ion. Un convertisseur dc-dc devait se charger de transformer la tension d’environ 11 volts de la batterie en tension utilisables pour le microcontrôleur et pour les autres périphériques inclus sur chaque véhicule.

La capacité de contrôler deux moteurs bdc par véhicule devait aussi être implémentée. La télécommande devait pouvoir contrôler chaque tank à l’aide d’une communication radiofréquence de 2.4 GHz. L’un des deux véhicules devait utiliser un gyroscope pour pouvoir gérer ses déplacements avec une précision accrue, et devait utiliser un sonar pour estimer les distances des murs lui faisant face.

Des ports d’extensions pour d’autres servos moteurs ou des LEDs, par exemple, devait aussi être disponibles.

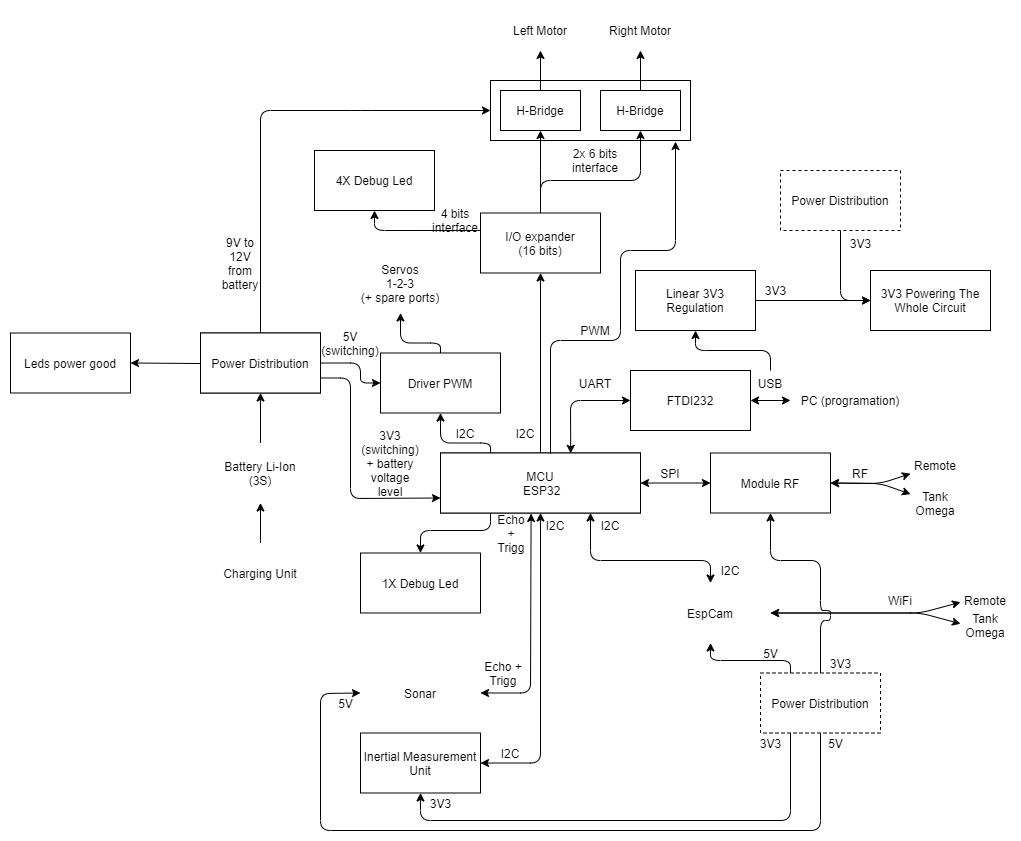
## Schéma bloc matériel du projet

Diagram

Description automatically generated

# Matériel

## Schémas



*Figure 1 : Schéma matériel d’Alpha*

## Fiches techniques

Les fichiers pdf des fiches techniques pour la partie Alpha du projet sont situées sur le git hub destiné à ce véhicule : <https://github.com/John-WL/Hydra-Alpha>

Note : certaines fiches techniques sont introuvables, donc voici leur description associée à un hyperlien qui mène vers digikey :

* Les condensateurs de 1µF smt : <https://www.digikey.ca/en/products/detail/w%C3%BCrth-elektronik/885382208005/9816358>
* Les résistances de 100 mOhms smt : <https://www.digikey.ca/en/products/detail/ohmite/PCS2512DR1000ET/10670884>
* Pour une liste non-exhaustive des composantes utilisées dans le projet : <https://trello.com/c/pOSYHtAN/64-composantes>

# Logiciel

## Programmes et projets des programmes

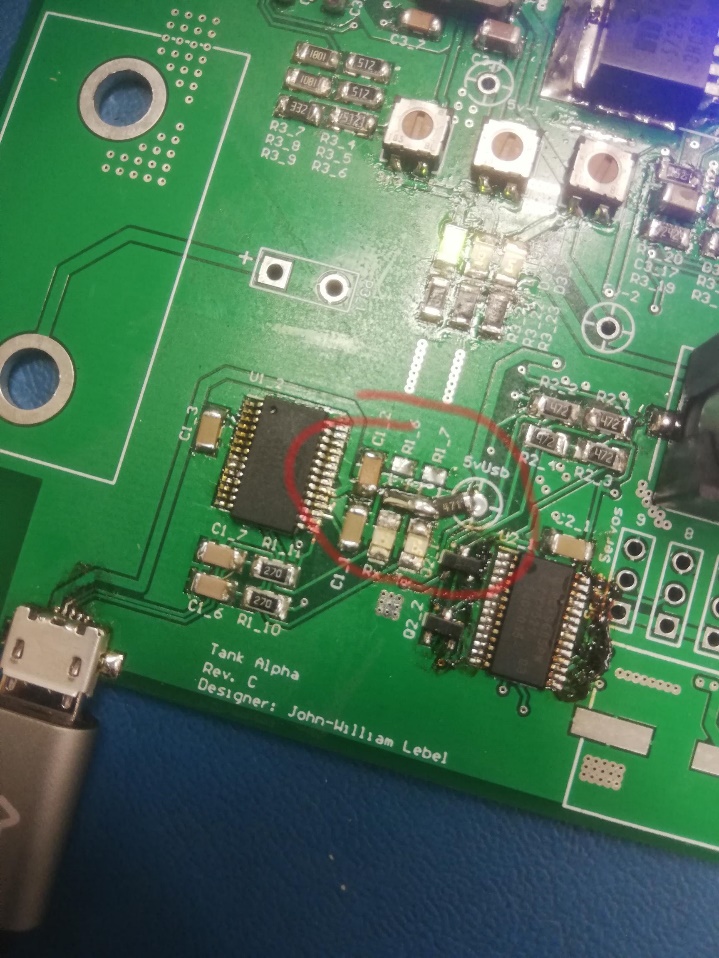
Tous les programmes de la partie Alpha ainsi que leur historique de développement sont accessibles sur git hub : <https://github.com/John-WL/Hydra-Alpha>

# Procédure de développement

## Procédure de déverminage de la carte et problèmes rencontrés

Plusieurs problèmes ont été rencontrés lors du montage de la carte électronique.

Le premier problème rencontré était l’alimentation des LEDs Rx et Tx qui servent à indiquer si une transmission est en cours sur le port série connecté à l’usb. Au lieu de connecter les LEDs sur le 5V, je les avais connectées sur la masse. Résultat : elles n’allumaient pas lors de la programmation. Après avoir constaté le problème, j’ai vite réalisé que j’avais, par chance, placé une borne de test pour l’alimentation 5V de l’USB juste à côté des LEDs de débogage. J’ai donc remplacé les deux résistances par une seule qui fait le pont entre la borne 5V et la LED la plus proche, et j’ai court-circuité l’anode des deux LEDs ensemble à l’aide d’une bonne grosse bulle d’étain.

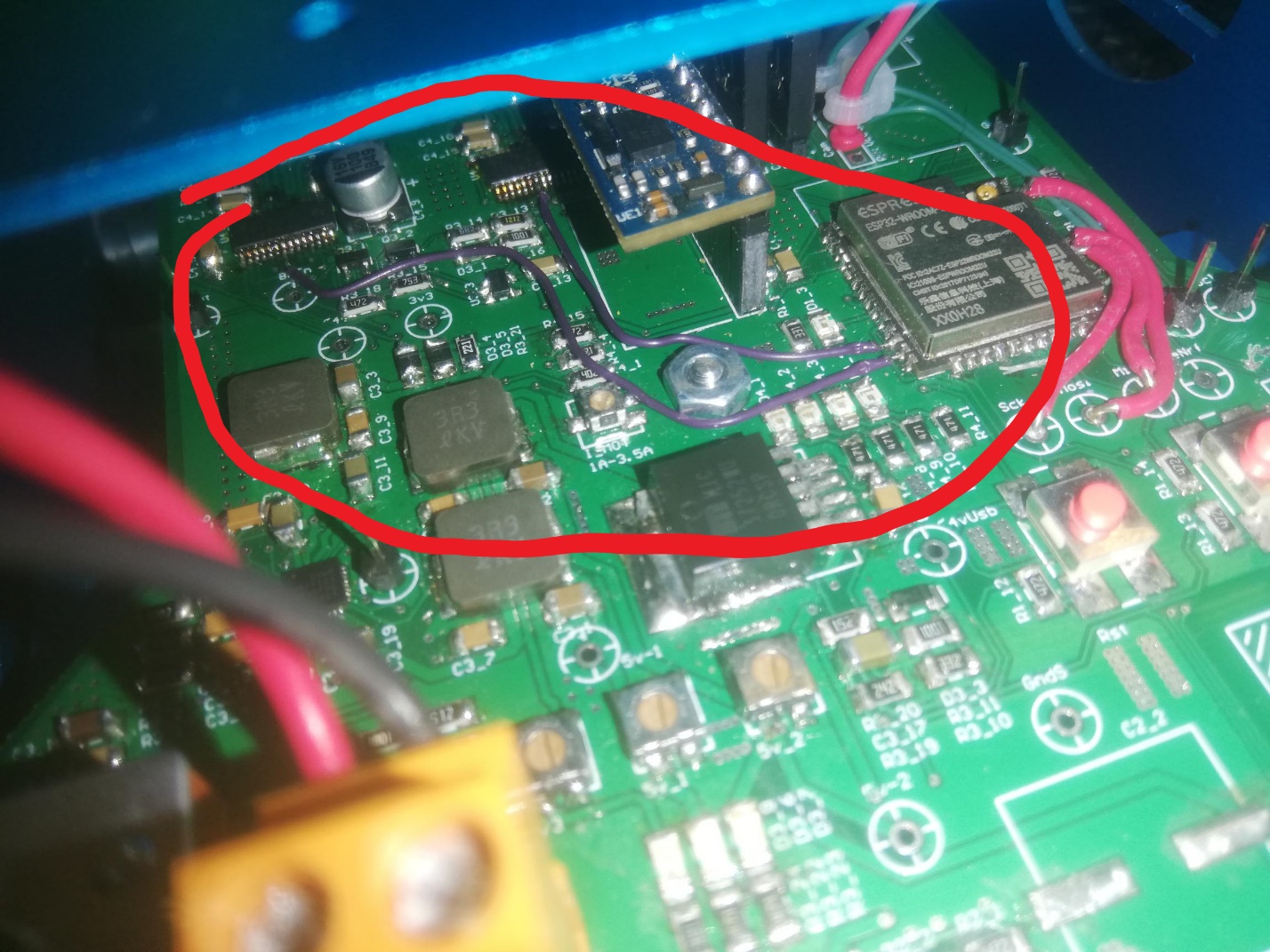


*Figure 2 : Déverminage des LEDs Rx et Tx*

Évidemment, en remplaçant les deux résistances par une seule de la sorte, on change un peu le fonctionnement du circuit. Si les deux LEDs sont allumées en même temps, on a deux fois plus de courant qui traverse la résistance. De ce fait, une plus grande tension sera présente à ses bornes lorsque les deux LEDs sont allumées en même temps que lorsqu’une seule LED est allumée à la fois, ce qui fait que la tension sera moins grande aux bornes des LEDs quand les deux sont allumées en même temps. Bref, avec ce nouveau circuit, les LEDs de programmation s’allument moins fort si elles consomment du courant simultanément. J’ai donc jugé qu’il était suffisant et acceptable de changer le circuit de cette façon.

Un autre problème intéressant que j’ai rencontré a été que les circuits intégrés servants à gérer les moteurs à balais n’avaient pas le fonctionnement qu’on pensait. Si on regarde rapidement la fiche technique (voici le liens digikey si ça vous intéresse : <https://www.digikey.ca/en/products/detail/texas-instruments/DRV8829PWPR/2674338>), on peut croire qu’il est possible de contrôler la vitesse d’un moteur à balais en sélectionnant le courant que le circuit donne en sortie à l’aide des pins I0 à I4. En effet, elles sont spécifiquement conçues pour sélectionner le courant qui circule dans le moteur. Le problème survient quand on se rend compte que cette sélection de courant n’est effective qu’au démarrage du moteur. On pensait donc pouvoir gérer la vitesse du moteur en faisant varier le courant qui y circule, car la vitesse du moteur est approximativement proportionnelle au courant qui y circule en fonctionnement normal (donc quand le moteur n’est pas bloqué et qu’il a un coefficient de friction approximativement proportionnel à sa vitesse), mais le circuit ne permettait en réalité pas du tout de gérer le moteur de cette façon. Si on essaie de gérer la vitesse du moteur uniquement avec les pins I0 à I4, la vitesse du moteur est incontrôlable. Au démarrage, le moteur commence effectivement doucement, mais le moteur accélère constamment, et se met rapidement à tourner à vitesse maximale.

La solution a donc été de couper les liens des pins « enable » de chaque circuit (car ces pins étaient connectés à l’alimentation 3.3 volts) et de souder de nouveaux liens entre ces pins « enable » et d’autres pins libres sur le microcontrôleur.



Ensuite, quand est venu le temps de programmer pour les moteurs à balais, au lieu de tenter de contrôler le couple des moteurs à l’aide des pins I0 à I4, un signal PWM est généré sur les pins « enable » des circuits. La direction des moteurs est encore contrôlée par le circuit d’extension d’IO qui servait à l’origine à interfacer les nombreux pins des deux circuits DRV8829.

Un autre problème rencontré fut la liaison entre l’EspCam et l’Esp32 du circuit principal. À l’origine, une liaison sur le port I2C avait été choisie pour pouvoir communiquer avec l’EspCam. Cette solution pourrait sembler adéquate à première vue, mais en codant la partie de l’EspCam, je me suis rendu compte de plusieurs problématiques en lien avec cette façon de communiquer avec l’EspCam :

* Les pins servant normalement à la communication I2C sont toutes les deux (sda et scl) cachées et utilisées par la caméra sur le circuit;
* L’Esp32 en général a étonnamment très peu de support pour la gestion du port I2C software en mode esclave;
* J’ai tenté de programmer par moi-même l’I2C software esclave sur l’Esp32. Mon code a fini par bien marcher, mais je n’arrivais pas à atteindre des vitesses très élevée (environ 20 KHz maximum). Considérant que d’autres périphériques utilisaient le même port I2C, cette diminution de vitesse aurait nettement affecté les autres circuits sur le même port, et je voulais éviter d’avoir à retester et redébugger cette partie du projet qui marchait déjà.

La solution choisie a donc été d’utiliser le port série (qui lui a ses deux pins de libres sur l’EspCam! wouhou!) à la place de L’I2C. Puisqu’on ne peut évidemment pas connecter un port série sur un port I2C sans causer de fâcheux problèmes majeurs de communications, j’ai décidé de connecter l’EspCam sur un des deux autres ports séries encore libres de l’Esp32 qui se situe sur le circuit principal. Ça donne quelque chose dans le genre (les signaux rx et tx correspondent aux deux fils vert):



Un autre problème majeur fut la connexion du module radiofréquence au microcontrôleur principal via le port SPI. Si on inspecte la fiche technique de l’Esp32 utilisé comme microcontrôleur principal, on se rend assez vite compte que certains pins semblent être dédiés au port SPI. Cependant, ces pins servent à la communication avec une carte SD par exemple. Ils sont en réalité inutilisables pour le module RF (avec les librairies d’arduino, en tout cas). J’ai donc dû couper les 3 traces principales (MOSI, MISO, CLK) pour les ressouder sur les bons pins sur le microcontrôleur (oups!). Voici à quoi ces rustines ressemblent (ce sont les fils en rouge. Ils sont plus gros car je n’avais pas de taille de fils plus petite chez moi) :



Le dernier problème majeur que j’aimerais aborder concerne mon convertisseur dc à dc. Après avoir soudé le convertisseur et après avoir constaté qu’il marchait parfaitement (il allait très bien), j’ai soudé les composantes restantes sur mon pcb. C’est seulement après avoir soudé la partie des servos moteurs qu’il s’est mis à osciller de façon extravagante. J’ai réalisé, après quelques heures de recherche, que le problème venait des condensateurs de découplage sensé servir à ce que le courant consommé par les servos moteurs n’affecte pas le reste du circuit logique. En fait, ces condensateurs de découplage se trouvent être aussi en parallèle avec le condensateur de découplage en sorti du convertisseur dc à dc. Ils débalançaient donc tout le circuit de puissance. Je n’ai pas d’image d’oscilloscope de prise, donc voici une photo de la mer avec de grosses vagues :



Le signal avait des oscillations crête à crête d’environ 1 à 2 volts, donc mon circuit ne cessait de redémarrer. J’ai donc simplement dessoudé les trois condensateurs de 1000 µF, et le signal est redevenu très lisse et très beau. Aussi, mes servos moteurs n’ont jamais causés aucun problème aux circuits logiques par la suite, donc les condensateurs me semblent être « inutiles », finalement. Je mets des guillemets, car je ne saurais dire si, sans ces condensateurs, l’alimentation 5V resterait effectivement belle si tous les ports « spare » de servos moteurs étaient utilisés en même temps.

## Liste des rustines

Voici la liste non-exhaustive des rustines effectuées durant le projet :

* J’ai brisé le port du connecteur de la batterie en tirant un peu trop fort dessus, donc j’ai dû faire une rustine à cet endroit pour la carte utilisée dans le projet. Cette rustine relève cependant d’une erreur de manipulation, et non d’une erreur de conception;
* Deux rustines ont été faites pour pouvoir mieux gérer les circuits de contrôle des moteurs à balais;
* Deux rustines ont été faites pour pouvoir changer le protocole de communication de l’EspCam;
* Trois rustines ont été faites car le mauvais port SPI avait été choisit pour la communication avec le module radiofréquence.

# Prototypes

## Liste de ce qui a été fait

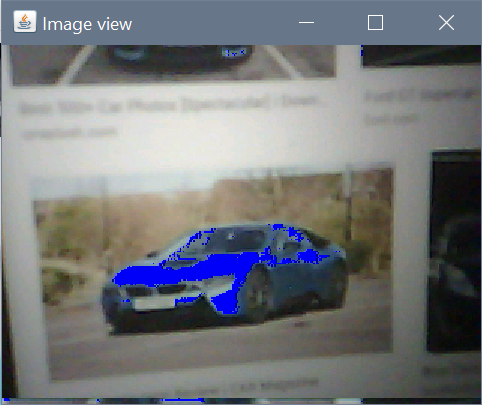
Voici une liste non-exhaustive de ce qui a été fait dans le projet :

* Tous les périphériques, à l’exception de la caméra, sont entièrement fonctionnels. Les ports de servos moteurs sont programmables et ajustables pour chaque servo moteur. Le sonar calcule plutôt bien la distance, à quelques millimètres de précision près. Il est possible de contrôler la vitesse des moteurs à balais ainsi que leur direction. L’unité de mesures inertielles fonctionne parfaitement bien (on peut connaître l’orientation du véhicule dans l’espace, on peut lire l’accélération et la vitesse angulaire du véhicule). Le module radiofréquence transmet et reçoit bien les données. La caméra arrive à transmettre des images par Wi-Fi;
* On peut contrôler le véhicule (les moteurs à balais ainsi que les servos moteurs) avec la télécommande. La télécommande est en mesure de recevoir les informations du véhicule, comme l’image ou le niveau de batterie, et elle est en mesure de les afficher à l’écran;
* On peut sélectionner les modes de fonctionnement, que ce soit autonome ou manuel.

## Liste de ce qui a été abandonné

Voici une liste non-exhaustive de ce qui a été abandonné durant le projet :

* L’utilisation d’un module GPS pour faire de la géolocalisation, car la gestion d’un tel module requiert une attention particulière au pcb, et la gestion de la force du signal avec une antenne soigneusement sélectionné rendaient cette partie du projet considérablement difficile à mettre en œuvre. De plus, c’était censé être un bonus pour si on avait le temps de l’implémenter, mais il faudrait que je refasse une autre carte électronique, ou que je connecte, à l’aide de rustines, un autre circuit développé à part, et on n’a pas vraiment eu le temps d’implémenter ce genre d’amélioration;
* La partie de détection de l’autre véhicule dans les images que la caméra prend a pris plus de temps que prévu. L’algorithme qui permet de sélectionner les pixels d’une couleur particulière dans l’image pour tenter d’en faire une moyenne marche, mais les paramètres, comme la couleur exacte recherchée, et le « range » de couleur acceptées ne sont pas encore déterminés de façon exacte. De plus, même si on trouvait les meilleurs paramètres possibles, je me demande à quel point l’algorithme marcherait, car les couleurs changent en fonction de l’orientation à l’éclairage et de la quantité de lumière dans une pièce. L’algorithme marche plutôt bien dans un environnement contrôlé, mais il y a beaucoup plus de bruit dans un environnement réel. Voici un exemple de ce que ça donne dans un environnement contrôlé (les pixels très bleus sont des pixels qui ont été modifiés pour montrer qu’ils ont été détectés comme étant de la bonne couleur. Dans cet exemple, l’algorithme recherchait des pixels de couleur bleu-clair) :



On voit bien que l’algorithme fait son travail. Cependant, il ne marche pas aussi bien dans un environnement moins contrôlé, comme dans une classe, par exemple. C’est très difficile à utiliser et à paramétrer. Nous n’avons donc pas eu le temps de l’implémenter dans le projet final;

* La gestion des mouvements du véhicule à l’aide de l’unité de mesures inertielles n’a pas été implémentée. Quelques tests ont été fait pour tenter de voir le potentiel d’une telle approche, mais les tests n’ont pas abouti à quelque chose d’utilisable dans la dernière version du projet. Les deux principaux modes de fonctionnements du véhicule sont le mode autonome et le mode automatique. Le mode automatique est géré par les joysticks de la télécommande, et l’autre véhicule n’utilise pas les mêmes capteurs inertiels. On voulait que les deux véhicules aient des comportements de conduite qui se ressemblent. De ce fait, nous avons conclus que ne pas utiliser d’unité de mesures inertielles permettrait d’obtenir des comportements de conduites qui sont très similaires pour les deux véhicules. C’est pourquoi l’unité n’a pas été utilisée pour gérer les déplacements dans le mode manuel. Le mode automatique, de son côté, requiert le fonctionnement complet de la caméra. On doit pouvoir détecter l’autre véhicule dans l’image pour que ce mode puisse fonctionner correctement. Puisque la caméra n’est pas complètement fonctionnelle pour le moment, l’utilisation de l’unité de mesures n’a pas pu être testé convenablement dans ce mode. L’unité de mesures inertielles, pour ces raisons, n’est pas utilisée du tout dans la dernière version présentée du projet.

## Liste des modifications à faire pour une prochaine version

Voici une liste non-exhaustive des modifications à faire pour une prochaine version du prototype :

* Enlever les trois condensateurs de découplage de 1000 µF sensés adoucir les alimentations des servos moteurs;
* Changer le port SPI du module radiofréquence sur le microcontrôleur principal;
* Modifier l’interfaçage des circuits de contrôle des moteurs à balais pour pouvoir adéquatement contrôler la vitesse des moteurs;
* Changer les connexions du port série de l’EspCam pour pouvoir souder correctement le header sur la carte;
* Changer l’alimentation des LEDs de programmation sur le 5V au lieu de sur la masse.

# Travail en équipe

## Répartition du travail en équipe

Jean-Sébastien s’occupait du véhicule Omega. John-William s’occupait du véhicule Alpha. Maxime s’occupait de la télécommande.

Jean-Sébastien s’est occupé de la communication radiofréquence pour l’équipe.

John-William s’est occupé de la communication Wi-Fi pour l’équipe.