Hes·so VALAIS WALLIS
Haute Ecole d'Ingénierie
Hochschule für Ingenieurwissenschaften

Hes·so
Haute Ecole Spécialisée
de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences and Arts
Western Switzerland

Filière Systèmes industriels

Orientation Infotronics

Travail de bachelor
Diplôme 2018

Jonathan Michel

Système embarqué pour drone Matrice 210

Professeur
Dr. Pierre-André Mudry

Expert
Dr. Pierre Huguenin

Date de la remise du rapport
24.08.2018

HES-SO Valais

SYND	ETE	TEVI
X	X	X

Données du travail de diplôme
Daten der Diplomarbeit

FO 1.2.02.07.CB
haj/11/01/2016

Filière / Studiengang **SYND**	Année académique / *Studienjahr* **2017/18**	No TD / *Nr. DA* **it/2018/71**
Mandant / *Auftraggeber* ☒ HES—SO Valais ☐ Industrie ☐ Etablissement partenaire *Partnerinstitution*	Etudiant / *Student* **Jonathan Michel** Professeur / *Dozent* **Pierre-André Mudry**	Lieu d'exécution / *Ausführungsort* ☒ HES—SO Valais ☐ Industrie ☐ Etablissement partenaire *Partnerinstitution*
Travail confidentiel / *vertrauliche Arbeit* ☐ oui / ja [1] ☒ non / nein	Expert / *Experte* *(données complètes)*	

Titre / *Titel*

Système embarqué pour drone Matrice 200

Description / *Beschreibung*

Le Matrice 200 de DJI est un drone permettant d'embarquer et communiquer avec un système électronique ad-hoc. Dans le cadre de ce travail de bachelor, l'objectif principal est de réaliser un système démonstrateur mettant en avant la communication entre un système de mesure et d'actuation embarqué avec le drone et de réaliser des mesures à distance, éventuellement en mode autonome.

Concrètement, les objectifs suivants sont visés :

1. Développement d'un système embarqué communiquant destiné à être déployé sur le drone.
2. Mise en place de la communication de données entre un système embarqué sur le drone et l'OS Android embarqué sur la télécommande.
3. Développement d'un logiciel Android permettant la mesure à distance de paramètres.
4. Si le temps le permet, déploiement d'un démonstrateur complet montrant l'acquisition autonome de mesure.

Signature ou visa / *Unterschrift oder Visum*

Responsable de l'orientation / filière
Leiter der Vertiefungsrichtung / Studiengang:

.....

[1] Etudiant / *Student* :

.....

Délais / *Termine*

Attribution du thème / *Ausgabe des Auftrags:*
16.05.2018

Présentation intermédiaire / *Zwischenpräsentation*
14 – 15.06.2018

Remise du rapport / *Abgabe des Schlussberichts:*
24.08.2018, 12:00

Expositions / *Ausstellungen der Diplomarbeiten:*
29, 30 – 31.08.2018

Défense orale / *Mündliche Verfechtung:*
04, 05 – 06.09.2018

[1] *Par sa signature, l'étudiant-e s'engage à respecter strictement la directive DI.1.2.02.07 liée au travail de diplôme.*
Durch seine Unterschrift verpflichtet sich der/die Student/in, sich an die Richtlinie DI.1.2.02.07 der Diplomarbeit zu halten.

Rapport reçu le / *Schlussbericht erhalten am* ..... Visa du secrétariat / *Visum des Sekretariats* .....

Travail de diplôme
| édition 2018 |

Filière
Systèmes industriels

Domaine d'application
Infotronics

Professeur responsable
Pierre-André Mudry
pierre-andre.mudry@hevs.ch

Système embarqué pour drone Matrice 210

Diplômant Jonathan Michel

Objectif du projet

Le but du travail de diplôme est de développer un système embarqué sur un drone afin de réaliser de manière autonome des mesures à distance et de les transmettre en temps réel au sol.

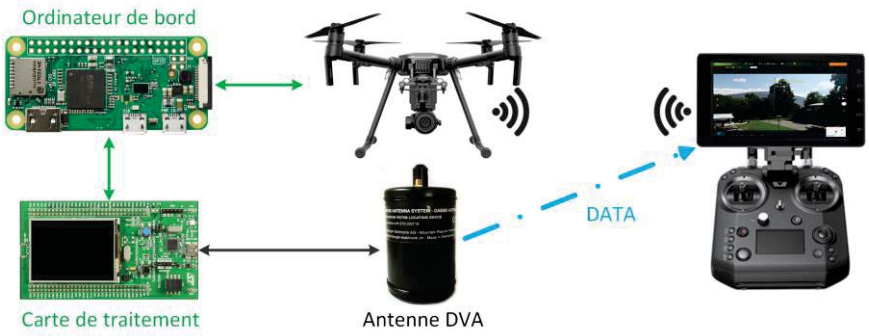
Méthodes | Expériences | Résultats

Le Matrice 210 est un drone spécialement destiné à une utilisation industrielle. Il dispose d'un port d'extension lui permettant de communiquer avec un ordinateur de bord afin d'être piloté.

Dans le cadre de ce travail de diplôme, nous avons embarqué une antenne de détection de victimes d'avalanche sous le drone et créé l'électronique et le logiciel nécessaires pour récupérer les valeurs de celle-ci en temps réel depuis le sol.

Pour réaliser cela, nous avons développé une application Android permettant de commander le drone de manière pseudo-autonome. Sur le drone lui-même nous avons mis en place le hard et le soft - sur Pi Zero W et STM32 - nécessaires à la lecture de l'antenne DVA ainsi qu'à la transmission en utilisant la communication fournie par le drone.

Les premiers tests ont montré que le système développé fonctionne comme prévu.



Un Raspberry Pi Zero W est utilisé comme ordinateur de bord. Il communique avec le drone pour le piloter et avec une carte STM32 qui traite le signal de l'antenne DVA. La valeur de l'antenne est ainsi renvoyée sur l'application Android en temps réel.

Table des matières

Première partie
Introduction

1 Préambule

Le *Matrice 210* est un drone à utilisation industrielle de *DJI*, leader mondial dans la fabrication de ces appareils. *DJI* met à disposition deux kits de développement (SDK - Software Development Kit). Le premier - *Onboard SDK* - est un kit de développement destiné à être utilisé sur une carte embarquée sur le drone afin de communiquer avec celui-ci. Le second - *Mobile SDK* - permet de créer des applications mobiles personnalisées sur un périphérique au sol. Ces deux kits combinés permettent de mettre en place des solutions industrielles personnalisées.

Ce travail de bachelor s'inscrit dans la démarche entrepreneuriale de Vincent Bontempelli et moi-même. Nous avons pour objectif de développer un drone autonome de sauvetage des victimes d'avalanches. Vincent est chargé de développer la recherche tandis que ce travail de bachelor concerne l'interfaçage avec le drone.

FIGURE 1: Matrice 210 [2]

2 Objectifs

L'objectif principal de ce travail de diplôme est de réaliser un système démonstrateur mettant en avant la communication entre un système de mesure embarqué sur le drone et l'utilisateur au sol afin de réaliser des mesures à distance. Le système de mesure n'est pas spécifié par le cahier des charges mais il s'agira de l'antenne utilisée pour la recherche des victimes d'avalanches.

Les tâches peuvent être séparées en quatre blocs présentés ci-dessous.

2.1 Système embarqué

Un système embarqué doit être déployé sur le drone. Il communiquera avec ce dernier en UART grâce au *Onboard SDK* afin de récupérer les informations nécessaires et transmettre des commandes à l'aéronef.

2. Source : `https://www.dji.com/matrice-200-series`

2.2 Communication Mobile-Onboard

Une communication de données doit être implémentée entre le système embarqué sur le drone et l'utilisateur au sol. Elle permettra d'échanger des données personnalisées entre ces deux points. La connexion sans fil LightBridge utilisée par DJI pour transmettre les informations entre la télécommande et le drone sert de canal de transmission. Les deux SDKs - Onboard and Mobile - disposent d'APIs permettant de mettre en place cette communication.

2.3 Application au sol

Une application Android doit être conçue afin d'interagir avec le système embarqué. Elle permettra de récupérer la valeur du capteur embarqué et de configurer l'ordinateur de bord pendant le vol. Pour ce faire, elle utilise le *Mobile SDK*.

2.4 Démonstrateur

Un système démonstrateur complet doit être mis en place afin de montrer l'acquisition de mesures à distance. Si le temps le permet, la démonstration peut présenter des fonctionnalités de vol autonome.

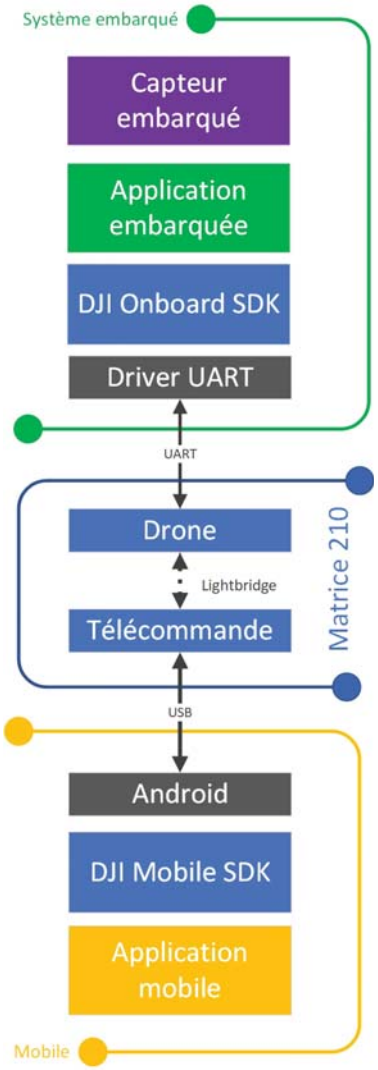


FIGURE 2: Schéma de principe

3 Planification

Une planification a été établie et est présentée ci-dessous. Elle est divisée en bloc d'une semaine et présente le déroulement général imaginé pour la réalisation du travail de diplôme. Certaines tâches seront exécutées en parallèle en fonction des besoins.

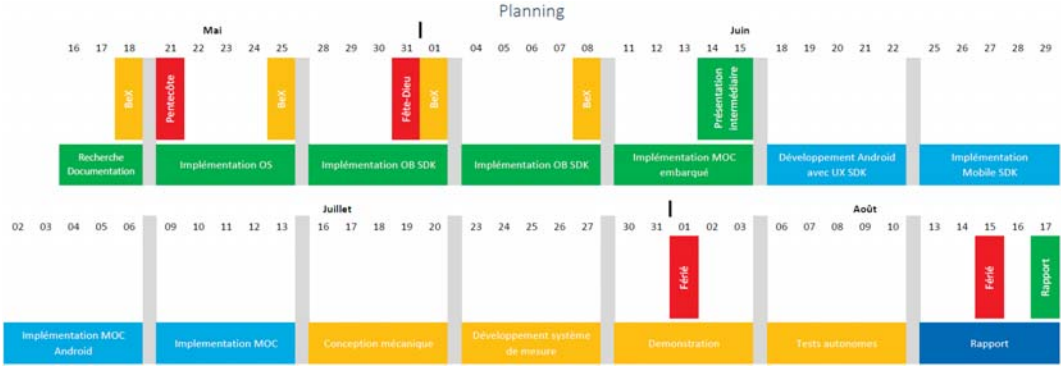


FIGURE 3: Planification

- **Semaine 1** : Recherches diverses et documentation
- **Semaine 2 à 5** : Développement de la carte embarquée : architecture logicielle, intégration du Onboard SDK, ...
- **Semaine 6 à 9** : Développement de l'application Android : conception de l'interface graphique, intégration du mobile SDK, implémentation de la MOC, ...
- **Semaine 10 à 11** : Intégration mécanique sur le drone et développement du système de mesure embarqué
- **Semaine 12 à 13** : Mise en place d'une démonstration, si le temps le permet de manière autonome
- **Semaine 14** : Semaine de réserve pour finalisation du rapport

A noter qu'une semaine a été ajoutée pour le rendu du rapport suite à ma participation au cours Business eXperience le semestre passé. En effet, durant le premier mois de travail de bachelor, les cours de ce module n'étant pas terminés j'étais absent les vendredis. La date de rendu du rapport a ainsi été décalée au 24 août 2018.

Les principales milestones du planning ont été respectées. Étant donné la nature du travail de bachelor, il était souvent de mise de travailler en parallèle sur le système embarqué et l'application Android dès la sixième semaine. Le montage mécanique et le développement du système de mesure ayant été partiellement réalisés par Vincent Bontempelli, le temps alloué était trop conséquent. Les dernières semaines ont ainsi majoritairement servi à développer et faire des tests de déplacements autonomes du drone. Le rapport a été rédigé durant les deux dernières semaines.

4 Structure du rapport

Ce rapport présente les différentes étapes de la réalisation du travail de bachelor. On y trouve :

— Un cahier des charges qui présente les objectifs du projet et les contraintes imposées.
— Une analyse du cadre légal concernant les technologies en jeu.
— Une présentation du matériel utilisé et des ressources à disposition.
— Le développement réalisé pour répondre à la problématique du cahier des charges.
— Une présentation des tests effectués et des résultats obtenus.
— Des propositions d'améliorations futures et une conclusion résumant l'ensemble.

Le glossaire, la table des figures et les références se trouvent en fin de document.

Les chapitres concernant les explications des différents SDKs sont inspirés de la documentation mise à disposition par DJI sur son site [3].

Sur la version numérique du rapport, les mots en *italique* sont cliquables et mènent rapidement vers la ressource concernée. Cette technique est utilisée lorsqu'un logiciel ou une entreprise connu·e est introduit·e pour la première fois. Les ressources spécifiques au projet sont quant à elle citées et disponibles en fin de document.

3. Site DJI Developer : `https://developer.dji.com/`

Deuxième partie
Analyse du contexte

5 Législation

Avant de commencer, il convient de faire un point sur la législation en vigueur pour l'utilisation de drones en Suisse. Il est autorisé de faire voler un drone sans autorisation pour les modèles dont le poids est égal ou inférieur à 30 kg MTOM *(Maximum Take-Off Mass)* à condition que le pilote maintienne un contact visuel permanent avec l'engin. Il n'est pas permis de voler au-dessus ou à moins de 100m d'un rassemblement de personnes - plus de 24 personnes [1].

Une restriction de vol existe pour les zones dans un rayon de 5km d'un aéroport ou d'un aérodrome. L'Office Fédéral de l'Aviation Civile OFAC met à disposition une carte interactive qui indique les zones dans lesquelles le trafic des drones est soumis à restrictions, voire interdit. [2].

DJI, de son coté, applique également des restrictions de vol sur ses appareils qui peuvent empêcher le décollage ou afficher des avertissements pendant le vol. Les restrictions dépendent des modèles et des pays, une carte est également disponible sur leur site [3].

Plus d'informations concernant l'utilisation des drones peuvent être trouvées sur le site de l'*Office Fédéral de l'Aviation Civile OFAC* [4].

5.1 Demande de vol spécial

Étant donné que la Haute Ecole d'Ingénierie de Sion est située à moins de 5km d'un aéroport, il a été nécessaire de faire une demande pour la coordination des activités aériennes spéciales afin de pouvoir faire voler le drone dans la cour de l'école. Cette dernière se fait auprès de *skyguide*, la société chargée des services de la navigation aérienne qui surveille l'espace aérien suisse. Elle doit être adressée au moins 10 jours ouvrés avant le jour de vol envisagé [5].

Dans le cadre de ce travail de bachelor, skyguide a accordé une pré-autorisation valide du 30 juillet au 7 septembre 2018 entre 8h et 18h. Elle couvre une zone d'un diamètre de 80 mètres pour une altitude maximale de 40 mètres autour de la cour du bâtiment A de l'école d'ingénieur. La zone est visible sur la Figure 4. Le but était de pouvoir réaliser des vols simples à basse altitude à proximité des laboratoires de travail.

Il est important de noter que skyguide délivre uniquement une pré-autorisation de vol. Il convient avant chaque vol de prendre contact avec la tour de Sion pour l'autorisation finale. La demande et la pré-autorisation sont disponibles en annexe pour plus de détails.

En pratique, la demande transmise à skyguide a été accordée en 2 jours ouvrés. Aucune demande n'a été refusée par la tour de Sion et il n'est pas nécessaire de la prévenir une heure avant, quelques dizaines de minutes suffisent.

A noter que du côté de DJI, la zone de restriction autour de l'aéroport de Sion est moins vaste. Ainsi il n'a pas été nécessaire de faire des démarches auprès du constructeur pour débrider le Matrice 210. C'est une procédure possible en cas de besoin.

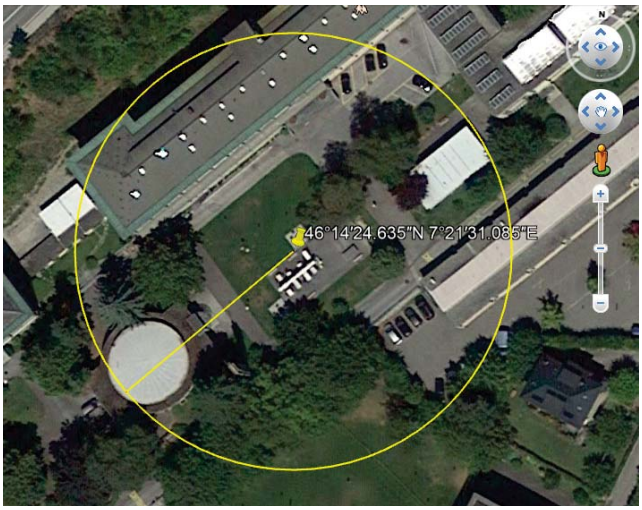


FIGURE 4: Zone concernée par la demande de vol spécial

5.2 Autorisation d'exploiter des drones à proximité d'un rassemblement

Comme indiqué précédemment, il n'est pas permis de voler au-dessus ou à moins de 100 mètres d'un rassemblement de personnes. Il a été envisagé de faire une demande auprès de skyguide pour faire une démonstration lors des expositions de travaux de bachelor.

Pour ce faire, il existe une *"Procédure d'autorisation simplifiée pour exploiter des drones et modèles réduits d'aéronefs captifs au-dessus et à proximité de rassemblements de personnes"* qui peut, entre autre, délivrer une autorisation *"SIDE"*. Cette dernière permet d'exploiter un appareil à moins de 100 mètres d'un rassemblement sans le survoler et concerne les aéronefs de moins de 30kg MTOM [6].

Il s'est avéré qu'une telle demande n'aurait pu être acceptée pour plusieurs raisons :
- En vue du nombre élevé de demandes, l'OFAC indique qu'il faut compter au minimum 3 mois pour traiter le dossier.
- La demande concerne les drones captifs, ce qui signifie que l'aéronef doit être relié au sol par un câble.
- La solidité du câble doit être démontrée en tentant, entre autre, de le sectionner avec les hélices du drone, ce qui aurait endommagé le matériel.
- Les autorisations ne sont délivrées qu'en cas de télépilotage, le vol autonome n'est pas envisageable.
- Le pilote doit prouver son aptitude à diriger le drone, par exemple en justifiant d'une formation adéquate ou de qualités aéronautiques établies par un carnet de vol.
- Finalement la demande est payante, elle peut coûter entre CHF 50.- et CHF 5000.- .

Pour ces raisons, cette demande n'a pas été entreprise. Les démonstrations seront réalisées sur simulateur pour les expositions de travaux de bachelor et en comité restreint sur le terrain.

5.3 Swiss U-Space

Finalement, skyguide a présenté cet été le *Swiss U-Space*. Il s'agit du premier système national de gestion du trafic des drones entièrement numérisé en Europe. Il a été développé en collaboration avec *AirMap*, le numéro 1 mondial des plateformes de gestion de l'espace aérien pour les drones [7].

Le but est de fournir aux drones un accès sûr et sécurisé au ciel suisse. Chaque aéronef enregistré et connecté au U-Space reçoit des informations en temps réel sur l'espace aérien et l'état du trafic. L'appareil quant à lui envoie à son tour des données sur sa position et sa trajectoire.

Le 26 mai 2018, Vincent Bontempelli et moi-même avons fait voler le Matrice 210 à Sierre afin de permettre à skyguide de tester et démontrer le système au *World Economic Forum* durant le *Drone Innovators Network*. Il s'agissait de faire un plan de vol sur l'application *AirMap* puis de faire voler le drone pendant la démonstration. Le résultat est visible ci-dessous.

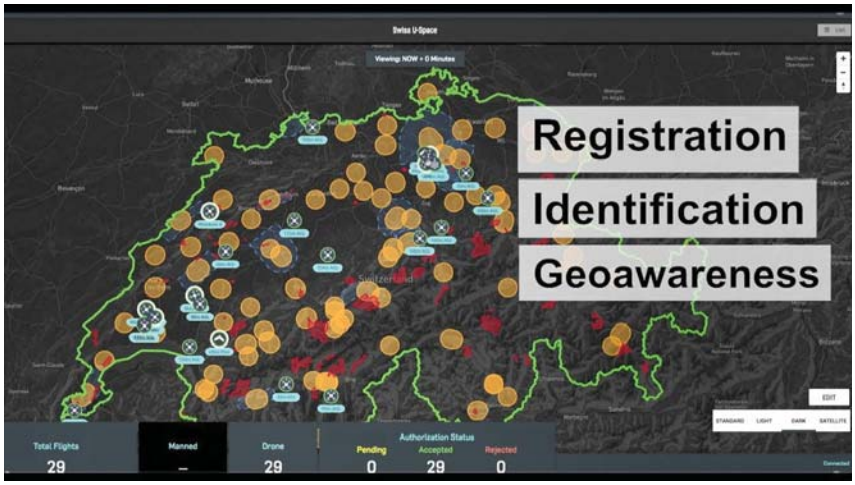


FIGURE 5: Tableau de bord du Swiss U-Space [2]

2. Source : https ://www.youtube.com/watch ?v=-dRj428sJfw

6 Drone Matrice 210

6.1 Drone

La gamme des *Matrice 200* est spécialement conçue pour le développement de solutions professionnelles. Elle est proposée par *DJI*, leader mondial dans la fabrication de drones.

Le modèle utilisé à l'école est le Matrice 210. Il s'agit du milieu de gamme de la série. Le modèle inférieur, le Matrice 200, ne peut pas être utilisé avec le Onboard SDK. Le modèle supérieur, le Matrice 210 RTK, dispose quant à lui d'extensions afin d'améliorer la précision du GPS.

Le Matrice 210 assure une portée de 3.5km pour 30 minutes de vol et peut soulever jusqu'à 2 kg[8]. Il est équipé d'une caméra *Zenmuse X4S*. Il s'agit d'une caméra 4k 60 FPS. Elle est fixée sous le drone sur une gimbal 3 axes stabilisée.

Les batteries utilisées sont les TB50 fournies avec le drone. Il en existe des plus puissantes, les TB55, mais elles ne sont pas disponibles à l'école. L'allumage de l'aéronef s'effectue avec un double appui prolongé sur le bouton à l'arrière de celui-ci.

Port d'extension
Le Matrice 210 dispose d'un port d'extension lui permettant de communiquer via une interface série afin d'échanger des informations et de le piloter. C'est sur ce port que sera connecté l'ordinateur embarqué. Un port USB permet de relier le drone à un ordinateur afin de le configurer et d'utiliser un simulateur de vol. Finalement, une sortie en tension permet d'alimenter l'ordinateur de bord.

Plus de détails sur le port d'extension et son utilisation sont disponibles dans le chapitre 11.2.

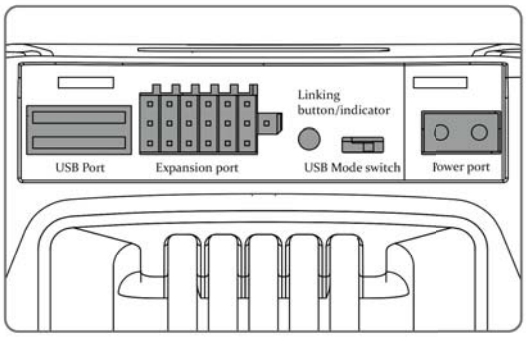


FIGURE 6: Porte d'extension à l'arrière du drone [3]

3. Source : Matrice 200 Series M210/M210 RTK User Manual V1.0 DJI ©, p.39

Vision System et Infrared Sensing System
Le Vision System permet au Matrice 210 de se repérer dans l'espace. Il est composé de deux caméras stéréo [A][C] et de deux capteurs à ultrasons [B] qui sont localisés à l'avant et sous le drone. Le Vision System utilise les images et les ultrasons afin d'aider le drone à maintenir sa position ce qui garantit une bonne stabilité en intérieur ou dans les environnements où le signal GPS est faible ou inexistant. De plus, le Vision System se charge de détecter les obstacles pour permettre au drone de les éviter.

Le Infrared Sensing System consiste en deux capteurs infrarouges [D] chargés de détecter les obstacles sur le dessus de l'appareil. Ils ont été désactivés pour permettre l'installation de l'ordinateur de bord sur le drone.

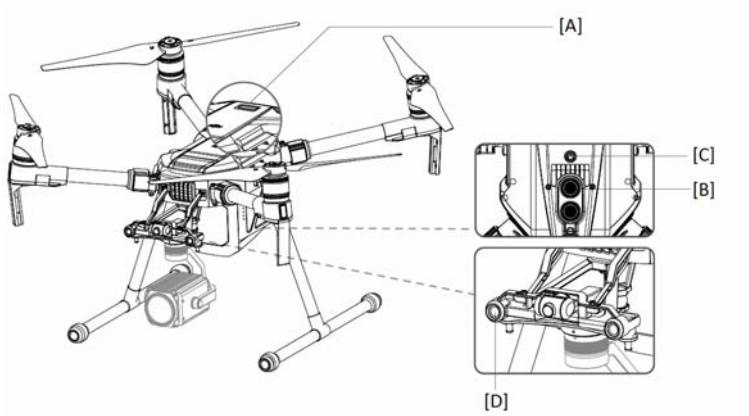


FIGURE 7: Vision System et Infrared Sensing System [4]

6.2 Télécommande Cendence

Le drone est pilotable avec la télécommande *Cendence*. Cette dernière utilise par défaut le mode 2 de pilotage. Celui-ci est normalisé dans l'utilisation d'aéronefs. Le fonctionnement est indiqué sur la Figure 8.

FIGURE 8: Télécommande Cendence et mode de pilotage [5]

Le drone dispose de trois modes de vols.
— P-mode (Positionning) - Ce mode fonctionne mieux lorsque le signal GPS est fort. Le drone utilise le GPS, le Vision System (ultrasons et images) et les capteurs infrarouges pour se stabiliser et éviter les obstacles.
— S-mode (Sport) - Les commandes du drone sont plus réactives. La vitesse maximale du drone passe de 61km/h à 83km/h. Dans ce mode, l'évitement d'obstacles n'est plus disponible.

4. Source : cf. 3, p.17
5. Source : cf. 3, p.46

— A-mode (Attitude) - Quand le GPS et le Vision System ne sont pas disponibles, le drone utilise uniquement son baromètre pour contrôler l'altitude.

Par défaut, le mode P est utilisé. Les modes de vol peuvent être changés avec le switch "P-S-A" à l'arrière gauche de la télécommande à condition d'activer cette option dans l'application DJI Go 4.

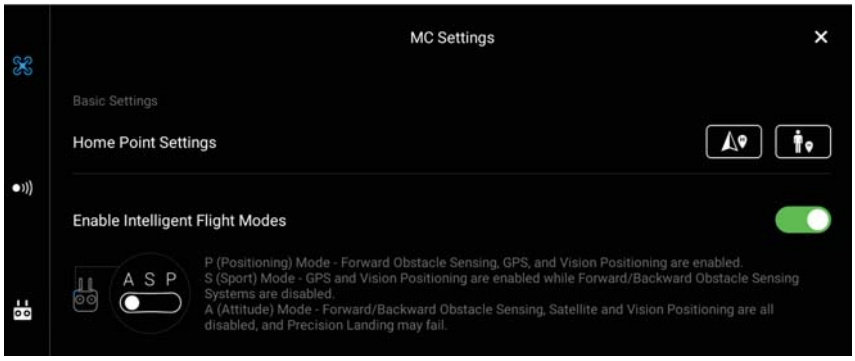


FIGURE 9: DJI Go 4 App - Activation des modes de vol

L'allumage de la télécommande s'effectue avec un double appui prolongé sur le bouton d'allumage. Plus de détails concernant l'utilisation de la télécommande et ses nombreux boutons peuvent être trouvés dans le mode d'emploi de la gamme des Matrice 200 [6].

6.3 Moniteur Crystal Sky

La télécommande est fournie avec un écran tactile *Crystalsky*. Ce dernier se branche directement sur le socle de la manette et tourne sur une surcouche basée sur Android Lollipop 5.1.1. Les deux applications habituelles pour piloter un drone de DJI - DJI Go 4.0 et DJI Pilot - sont installées par défaut mais il est tout à fait possible d'y installer une application externe. C'est sur ce support que sera développé l'application chargée de récupérer les données du capteur embarqué sur le drone.

La version actuelle de la surcouche DJI est la V2.06.06.00.

FIGURE 10: DJI Crystal Sky [7]

6. Mode d'emploi : Matrice 200 Series M210/M210 RTK User Manual V1.0 DJI ©
7. Source : `https://www.dji.com/crystalsky`

6.4 Logiciels et applications

DJI met à disposition plusieurs logiciels et applications pour l'utilisation et le développement de leurs produits.

6.4.1 DJI Assistant 2

Le logiciel DJI Assistant 2 permet de mettre à jour les contrôleurs de vols de DJI, de télécharger les données de vol, de calibrer les capteurs de détection d'obstacles et de configurer le Onboard SDK. Pour l'utiliser, le drone doit être relié par USB à l'ordinateur et le switch de mode USB à l'arrière du drone doit être sur la position gauche.

Le logiciel est disponible sur PC et MAC. La version actuellement utilisée est la V.1.2.4.

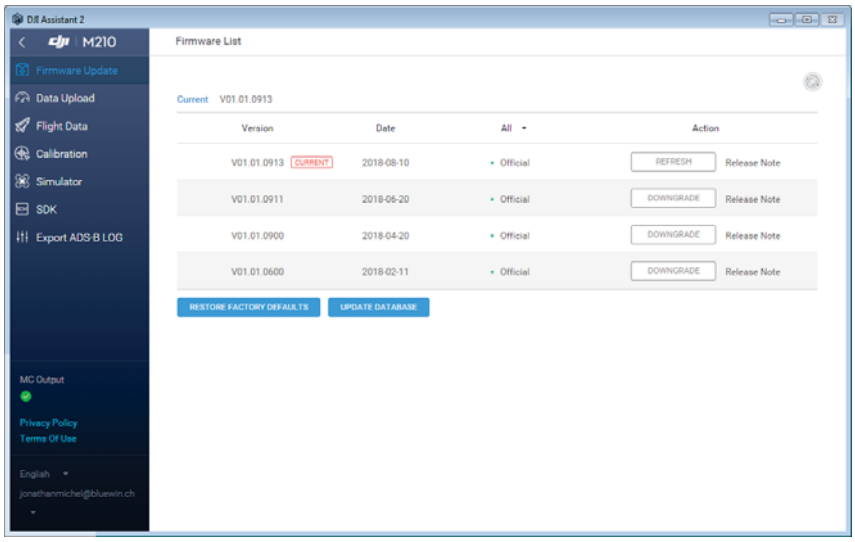


FIGURE 11: DJI Assistant 2

Un simulateur de vol est disponible dans le logiciel pour tester les commandes envoyées au drone. Pour l'utiliser, le drone doit être allumé, hélices retirées et connecté à l'ordinateur de bord comme dans une configuration habituelle.

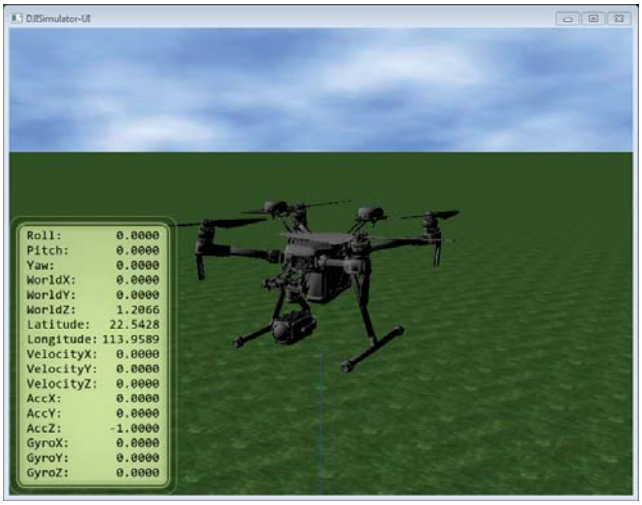


FIGURE 12: DJI Assistant 2 - Simulateur

6.4.2 DJI Go 4 App

DJI met à disposition une application à utiliser pour piloter le drone. Elle permet d'avoir un retour vidéo, de contrôler la gimbal, la caméra et d'autres fonctionnalités liées au drone. La version utilisée est la V4.2.21.

FIGURE 13: DJI Go 4 App[8]

6.4.3 DJI Pilot

Une seconde application dédiée au pilotage du drone est installée sur le moniteur Crystal Sky. Elle propose une interface "Manual Flight" similaire à celle de l'application DJI Go 4 et une interface "Mission Flight" pour réaliser des missions. Cette application est encore en beta V0.6.1 et l'interface pour les missions n'est à ce jour pas disponible.

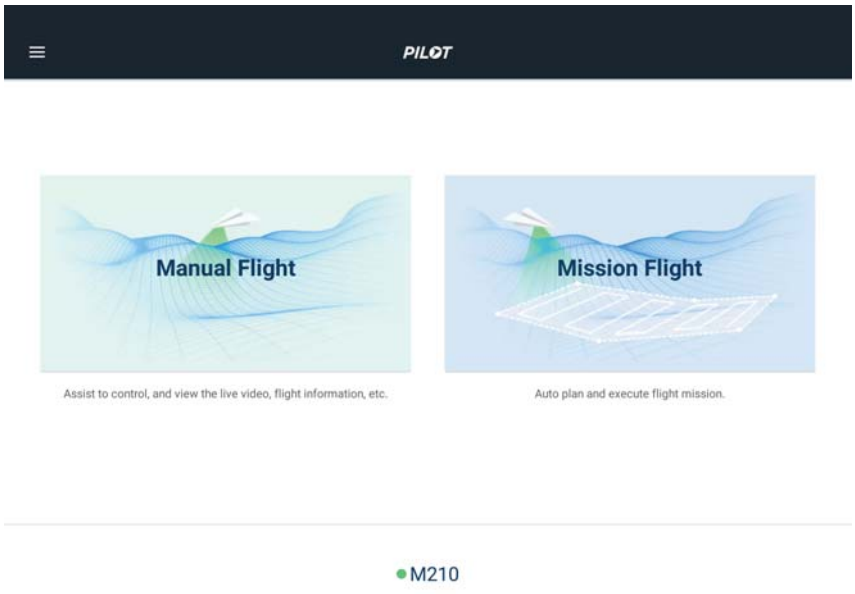


FIGURE 14: DJI Pilot App

8. Source : https://www.youtube.com/watch?v=RxxVibQKDSY

6.4.4 DJI Bridge

Lors du développement d'une application Android avec le Mobile SDK, il est nécessaire de relier le périphérique mobile par USB à la télécommande du drone. Cela signifie que le débogage de l'application ne peut que se faire que par Wifi. Bien que fonctionnelle, cette solution est lente car le profilage ou le transfert d'une nouvelle version sur le périphérique mobile sont des tâches gourmandes en ressources qui prennent du temps.

Afin de contourner ce problème, DJI met à disposition une application *DJI Bridge* qui doit être lancée sur un périphérique mobile connecté à la télécommande. L'application communique via le réseau avec un second périphérique qui exécute l'application basée sur le Mobile SDK.

Avec cette solution, il est ainsi possible de connecter le périphérique sur lequel tourne l'application basée sur le Mobile SDK à un PC par USB ou également d'utiliser l'émulateur Android d'*Android Studio*.

Le fonctionnement de l'application est détaillé sur son Github [9]. Il est nécessaire que les deux périphériques mobiles soient connectés au même réseau Wifi. Dans le cadre de ce travail de bachelor, le réseau *secure-hevs* de l'école ne convient pas car les connexions entre périphériques en local sont probablement bloquées. La solution a été de se servir du partage de connexion réseau du portable connecté à la télécommande.

FIGURE 15: DJI Bridge

9. DJI Bridge Github : `https://github.com/dji-sdk/Android-Bridge-App`

7 Onboard SDK

7.1 Vue d'ensemble

Le *Onboard SDK* (OSDK) permet d'étendre les capacités du drone. Il s'agit d'une librairie open source qui permet aux systèmes embarqués de communiquer avec les contrôleurs de vols et les drones de DJI via une interface série. Le OSDK donne accès à des informations de télémétrie, de contrôle de vol et à d'autres fonctionnalités permettant à un développeur de relier un ordinateur de bord au drone afin de le piloter. Il est principalement utilisé lorsque des calculs doivent être faits sur le drone suite à l'acquisition d'un capteur embarqué, par exemple.

La version utilisée lors de ce travail de bachelor est un fork [10] issu de la version 3.6 proposée par DJI sur leur Github [11]. Cette version contient deux bugfixes de la librairie officielle.

Le OSDK inclut :
— Une librairie open source en C++ pour piloter le drone via l'interface série
— Des codes exemples et des tutoriels
— Une documentation d'API

La plupart des fonctionnalités des produits DJI sont accessibles via le OSDK. Les développeurs peuvent automatiser le vol, contrôler la caméra et la gimbal, récupérer l'état de différents composants, ...

Contrôle de vol
Il est possible de piloter le drone de plusieurs façons :
— Position Control - Contrôler la position du drone
— Velocity Control - Contrôler la vélocité du drone
— Attitude Control - Contrôler l'attitude du drone
— Angular Rate Control - Contrôler le taux d'attitude du drone
— Mission - Des itinéraires prédéfinis peuvent être suivis.
Ces différents contrôles sont présentés plus en détail dans le chapitre 11.10.2.

Contrôle de la caméra et de la gimbal
Le Onboard SDK permet d'interagir avec la caméra et la gimbal du drone.
— Caméra - Prise de vidéos et de photos
— Gimbal - Contrôle de la position et de la vélocité de la gimbal

Synchronisation hardware
Il est possible de programmer un signal de synchronisation hardware sur les contrôleurs de vol et les drones DJI. Cette option propose les fonctionnalités suivantes :
— Harware Pulsed Signal - Génère une impulsion sur le port d'extension qui peut être utilisée par l'ordinateur de bord ou par des capteurs externes.
— Software Data Packet - Synchronise les données de l'IMU et les timestamps en fonction de l'impulsion hardware.

Contrôle MFIO
Le OSDK permet de configurer les pins des Multi-Function IO du port d'extension du drone. Les fonctionnalités disponibles sont les suivantes :
— Sortie PWM
— Convertisseur analogique-digital
— GPIO

Télémétrie
De nombreuses informations sur les capteurs et le statut du drone peuvent être récupérées jusqu'à une fréquence de 200Hz.

10. Fork du DJI Onboard SDK : `https://github.com/jonathanmichel/Onboard-SDK/tree/dev`
11. Github DJI Onboard SDK : `https://github.com/dji-sdk/Onboard-SDK`

Données des capteurs
— GPS
— RTK
— Boussole
— Baromètre

Statut du drone
— Vélocité
— Altitude
— Position de la gimbal
— Niveau de batterie
— Quaternions
— Accélération linéaire
— Vitesse angulaire

Plus d'informations sur le fonctionnement de la télémétrie sont disponibles au chapitre 11.9.

Caméra stéréo

Les développeurs peuvent accéder aux données de la caméra frontale et de celle en dessous du drone.
— Niveau de gris de la caméra frontale en résolution VGA ou QVGA
— Carte de disparité de la caméra frontale
— Niveau de gris de la caméra inférieure en qualité QVGA

7.2 Plateformes

7.2.1 Possibilités

Le *OSDK* est disponible sur plusieurs plateformes de développement :

FIGURE 16: DJI Onboard SDK - Plateformes de développement [12]

Linux

L'avantage d'une solution sous Linux est de pouvoir utiliser les fonctionnalités d'un OS (multithreading, queues, ...) et d'assurer une meilleure portabilité pour les projets futurs au sein de la HEI si ceux-ci nécessitent une puissance de calcul supplémentaire.

Linux est disponible sur un grand nombre de périphériques. *Intel* propose différentes cartes de développement, parmi lesquelles la *Intel Edison* par exemple. Cette dernière a déjà été utilisée au sein de l'école mais n'est plus produite actuellement. Il existe également la carte *Aero* ou encore la *Jetson TX2* de *Nvidia* plutôt utilisée pour des applications dans l'IA. Finalement un *Raspberry Pi* peut être envisagé, malgré une utilisation habituellement dans des projets de domotique plutôt qu'en embarqué.

Il faut être attentif qu'avec ces solutions il n'est pas possible de mettre en place une solution temps réel, bien qu'il soit concevable de s'en rapprocher en appliquant une patch comme *PREEMPT-RT*.

12. Source : `https://developer.dji.com/onboard-sdk/`

ROS

ROS (Robot Operating System) est une solution modulaire permettant de mettre en place des projets de robotique. Elle est très utilisée en automatisme, ses points forts étant l'existence de nombreux modules prêts à l'emploi permettant de mettre en place facilement ce type de projet. Il s'agit d'une solution tournant sur Ubuntu. Elle semble inconnue au sein de l'institut de la HEI.

Qt

Le principal avantage lors de l'utilisation de la librairie *Qt* est d'assurer une portabilité par simple recompilation du code source sur la plateforme cible. Elle est parfaitement adaptée pour le développement d'interfaces graphiques ou d'une architecture complexe pour une application fonctionnant sur le mécanisme des signaux-slots par exemple. Elle met à disposition de nombreux composants, ce qui la rend très pratique à utiliser mais également un peu lourde pour de l'embarqué.

STM32

STM32 est la plateforme la plus bas niveau des quatre ci-dessus, elle permet un contrôle total du comportement du système, ce qui est une bonne chose pour la sécurité. La puissance de calcul sur un STM32 est plus faible que sur une carte *Jetson TX2* par exemple, mais les tâches à effectuer ne demandent pas forcément une puissance particulière. Il s'agit d'implémenter une communication UART, de traiter des données et d'exécuter des machines d'état. Un STM32 peut tout à fait être adapté.

7.2.2 Choix

Dans un premier lieu il a été choisi de développer sur STM32 avec une carte Nucleo-L476RG. Une solution sur Raspberry Pi a ensuite été mise en place. Plus d'informations sur les raisons qui ont influencé ce choix sont disponibles au chapitre 11.

7.3 Déploiement

Le diagramme de la figure 17 illustre comment le Onboard SDK interagit avec l'application utilisateur et le drone.

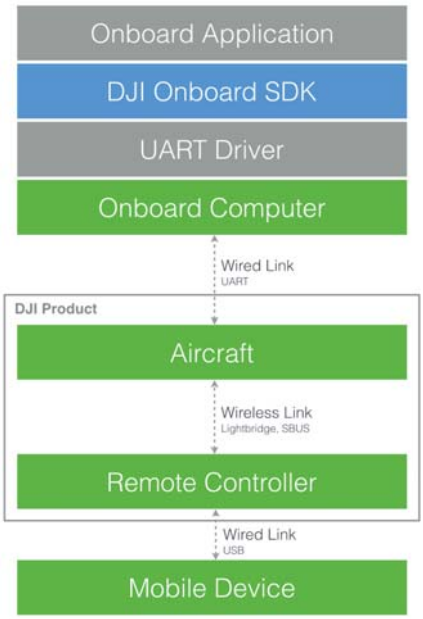


FIGURE 17: Schéma de principe [13]

8 Mobile SDK

8.1 Vue d'ensemble

Le *Mobile SDK* (MSDK) est un kit de développement software qui permet aux développeurs d'accéder aux drones de DJI . Le MSDK simplifie le développement en gérant les fonctionnalités de bas niveau tel que la gestion de la batterie, la transmission de signal et la communication avec le drone.

Le MSDK inclut :
— Une librairie/framework qui donne accès aux produits DJI
— Des codes exemples et des tutoriels
— Une documentation d'API

La plupart des fonctionnalités des produits DJI sont accessibles via le MSDK. Les développeurs peuvent automatiser le vol, contrôler la caméra et la gimbal, télécharger les médias stockés sur le drone, récupérer le flux vidéo en temps réel ainsi que les valeurs des capteurs, ...

Contrôle de vol
Il est possible de piloter le drone de trois façons :
— Manuellement - Le pilote commande le drone avec la télécommande pendant que le MSDK récupère les informations vidéo et les valeurs des capteurs.
— Joystick virtuel - Le MSDK permet de générer virtuellement les commandes d'un joystick afin d'émuler un pilote.
— Missions - Des itinéraires prédéfinis peuvent être suivis.

Caméra
La caméra et la gimbal sont programmables. Il est possible d'interagir sur plusieurs paramètres :
— Mode de caméra - Photo ou vidéo
— Exposition - Obturateur, ISO, compensation de l'ouverture et de l'exposition
— Paramètres de l'image - Ratio d'image, contraste, teinte, netteté, saturation et filtres
— Paramètres de vidéos - Résolution et fréquence d'image
— Direction - Positionnement de la gimbal

Live vidéo
Il est possible de récupérer le flux vidéo de la caméra principale du drone, même quand la caméra enregistre des photos et vidéos sur le stockage du drone.

Données des capteurs
Les données de plusieurs capteurs peuvent être récupérées via le MSDK. On peut par exemple récupérer la position GPS, les données de la boussole et du baromètre, la vélocité et l'altitude du drone à une fréquence maximum de 10Hz.

Télécommande, batterie et liaison Wireless
Les informations relatives à la télécommande, au niveau de batterie et à la connexion Wireless peuvent être récupérées.

13. Source : `https://developer.dji.com/onboard-sdk/documentation/introduction/onboard-sdk-introduction.html`

8.2 Plateformes

Le MSDK est disponible sur iOS dès la version 9.0 et sur Android dès la version 5.0.0.

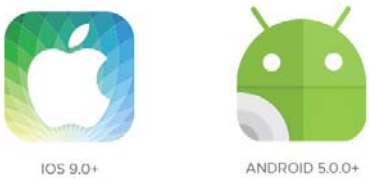


FIGURE 18: DJI Mobile SDK - Plateformes de développement [14]

Dans le cadre de ce travail de bachelor, l'application sera développée sur Android sur le moniteur tactile Crystal Sky présenté au chapitre 6.3.

8.3 Déploiement

Le diagramme de la figure 19 illustre comment le Mobile SDK interagit avec l'application utilisateur et le drone.

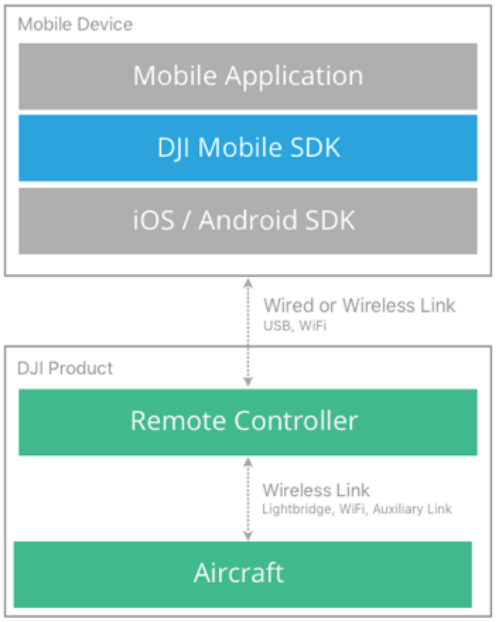


FIGURE 19: DJI Mobile SDK - Plateformes de développement [15]

14. Source : https://developer.dji.com/mobile-sdk/
15. Source : https://developer.dji.com/mobile-sdk/documentation/introduction/mobile_sdk_introduction.html

8.4 User eXperience SDK

La plupart des applications dédiées au contrôle d'un produit DJI utilisent des fonctionnalités de base communes comme :

— Afficher une vue en live du retour vidéo
— Afficher l'état du drone - niveau de batterie, puissance du signal, position, ...
— Proposer des panneaux de configuration pour les paramètres du drone
— Mettre à disposition des interfaces pour les fonctionnalités automatiques de base - décollage, atterrissage, return to home (RTH), ...

Lors de la création d'une application, un développeur doit mettre en place ces fonctionnalités avant d'en ajouter des personnalisées.

Pour ce faire, DJI met à disposition un *UX (User eXperience) SDK* (UXSDK) qui fournit les éléments graphiques de ces principales fonctionnalités. Il permet de créer une vue similaire à celle présentée à la Figure 20.

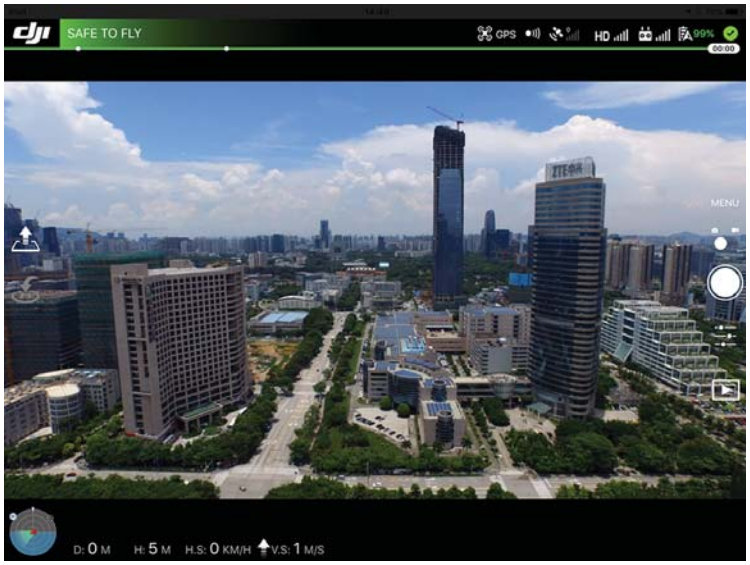


FIGURE 20: DJI User eXperience SDK [16]

Les éléments sont regroupés en trois catégories principales : Widgets, Collections et Panels. Ils peuvent être simplement ajoutés à l'application sans se soucier de leur fonctionnement car ils sont liés au MSDK et se mettent à jour automatiquement.

16. Source : https://developer.dji.com/mobile-sdk/documentation/introduction/ux_sdk_introduction.html

Widget

Les widgets sont les éléments de base du UXSDK. Ils représentent typiquement un état ou permettent d'exécuter une action, comme un indicateur de batterie ou le bouton de décollage automatique.

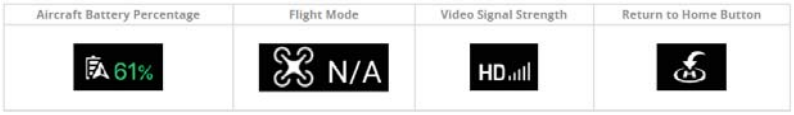


FIGURE 21: DJI User eXperience SDK - Widgets [17]

Collections

Une collection regroupe plusieurs widgets d'un même type organisés dans un ordre prédéfinis. Il peut s'agir par exemple d'une barre de statut. Les collections sont uniquement disponibles sur iOS.

FIGURE 22: DJI User eXperience SDK - Collection [18]

Panels

Les panels sont les éléments les plus complexes. Ils contiennent plusieurs informations ou paramètres. C'est le cas par exemple du menu des paramètres de la caméra ou de la checklist de vol.

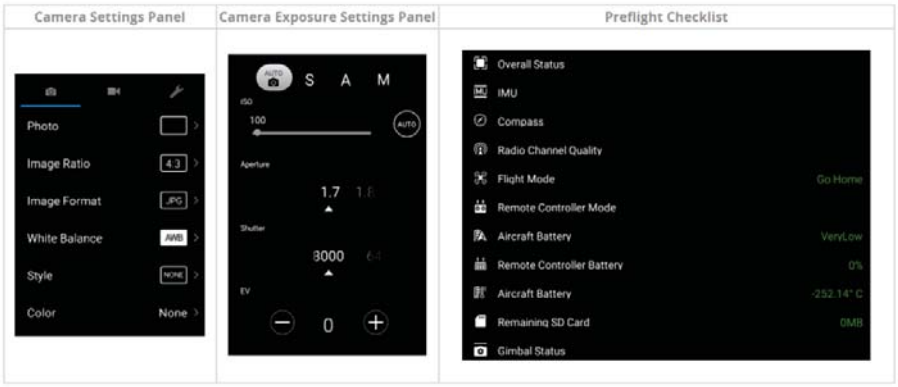


FIGURE 23: DJI User eXperience SDK - Panels [19]

17. Source : https://developer.dji.com/mobile-sdk/documentation/introduction/ux_sdk_introduction.html
18. cf. 17
19. cf. 17

9 Comparaison entre les deux SDK

Les deux SDK présentés - Onboard et Mobile - permettent de créer des applications pour contrôler les produits DJI mais sont optimisés pour différentes utilités et plateformes. Le Mobile SDK est destiné à être déployé sur un périphérique mobile et se connecte au drone grâce à la connexion sans fil de la télécommande. Le Onboard SDK est à utiliser sur un ordinateur Linux ou un STM32 et est connecté directement au drone via une interface série.

Le tableau ci-dessous compare les deux SDK.

Catégorie	**Onboard SDK**	**Mobile SDK**
Plateforme	Linux, STM32	Android, iOS
Langage	C++	Java, Objective C, Swift
Connexion avec le drone	Filaire (Série)	Sans fil (Lightbridge)
Fréquence télémétrie max	200 Hz	10 Hz
Missions disponibles	Waypoint, Hotpoint	Waypoint, Hotpoint, Follow Me, ActiveTrack, TapFly, Timeline
Télécommande nécessaire	Non	Oui
Contrôle de la gimbal	Oui	Oui
Contrôle de la caméra	Limité	Complet

TABLE 1: Comparaison Onboard SDK et Mobile SDK

Il existe également des différences dans les produits DJI supportés par les deux SDK mais tous deux sont disponibles sur le Matrice 210. Au-delà des points cités ci-dessus les SDK sont destinés à des applications différentes. Le OSDK est à utiliser lorsque le développeur veut par exemple :

— Suivre des trajectoires précises
— Voler sans la télécommande
— Utiliser un capteur ou un système de communication externe pour piloter le drone

Le Mobile SDK quant à lui est plus facile à mettre en place et propose des fonctionnalités plus avancées pour le contrôle de la caméra et les missions.

10 Mobile-Onboard Communication

Le Onboard SDK et le Mobile SDK peuvent être utilisés simultanément, tous deux disposent d'API pour échanger des données personnalisées. La connexion Lightbridge existante entre la télécommande et le drone est utilisée comme canal de transmission. Cette fonctionnalité permet par exemple d'envoyer les données mesurées par un capteur embarqué sur l'application au sol ou à l'inverse de transmettre des ordres à l'ordinateur de bord depuis le sol. Elle garantit un transfert en up (Mobile vers Onboard) d'approximativement 1kB/s pour 8kB/s en down (Onboard vers Mobile)[9].

Troisième partie
Développement réalisé

11 Système embarqué

11.1 Tests sur STM32

Dans un premier temps, il a été envisagé d'utiliser un RTOS sur STM32. Bien que les tâches cruciales comme la stabilisation, l'évitement d'obstacles, etc... soient réalisées par le drone, il est toujours agréable de disposer d'un OS pour créer une application bien structurée et dont la maintenance est facilitée. Le RTOS n'est donc pas directement choisi pour son déterminisme. Le choix du RTOS s'est tourné vers *Zephyr OS*. Il s'agit d'un système d'exploitation open-source léger, conçu pour les appareils aux ressources limitées et supportant plusieurs architectures. Il est actuellement en développement et fait partie de la *Linux Foundation*. La carte *Nucleo-L476RG* a été choisie pour son support avancé de Zephyr. Les détails du choix du RTOS et de la carte se trouvent en annexe.

L'implémentation de Zephyr s'est bien déroulée mais au moment d'ajouter la librairie C++ du Onboard SDK il a fallu faire face à de nombreux conflits d'implémentation de types et de méthodes entre la toolchain, le kernel de Zephyr et le Onboard SDK. Le problème venait des différences de langage entre le kernel en C de Zephyr et la librairie C++ de DJI. Cette différence était connue au moment du choix de Zephyr mais la cohabitation s'est révélée plus fastidieuse que prévu, Zephyr indiquant pourtant supporter le C++[10]. Après plusieurs essais avec différentes toolchains et des discussions avec Messieurs Michael Clausen et Marc Pignat, il s'est avéré qu'il n'était pas pertinent de continuer dans cette direction. Le code de cette solution est néanmoins disponible sur Github [1].

Afin de tester le Onboard SDK, une solution a été mise en place sur *SW4STM32 (System Workbench for STM32)* à l'aide de l'outil STM32CubeMX. L'utilisation de la carte Nucleo précédemment citée a été conservée. Le code de cette solution est disponible sur Github [2]. Bien que fonctionnelle, cette solution n'était pas très élégante. En effet, *SW4STM32* est disponible uniquement sur Windows et le projet généré par STM32CubeMX est relativement mal structuré. Il ne permet d'ailleurs pas de générer un code basé sur CMakeLists mais uniquement un projet pour *SW4STM32* - ou d'autres IDE similaires - ce qui est un coup dur pour la portabilité. De plus, l'implémentation d'un OS a ainsi été mise de côté.

11.2 Solution sur Raspberry Pi

Les tests sur STM32 ont permis de se rendre compte qu'une solution sous Linux est plus adaptée. Elle permet de proposer une base facilement réutilisable pour des projets plus ambitieux à l'avenir au sein de la HEI. De plus, la disponibilité de fonctionnalités liées à un OS (multithread, queues, système de fichiers, ...) est agréable. Il s'agit donc d'une solution plus propre et avec une portabilité accrue.

Suite à la présentation intermédiaire et sur les conseils de Messieurs Marc Pignat et Pierre-André Mudry, il a été choisi de se tourner vers un Raspberry Pi. Des premiers tests ont été faits sur un *Raspberry Pi 3 Model B+* disponible à l'école avant de se tourner vers un *Raspberry Pi Zero W* plus compact.

1. Tests avec Zephyr OS : `https://github.com/jonathanmichel/zephyr_os`
2. Solution sur Nucleo : `https://github.com/jonathanmichel/pilot-sdk-nucleo`

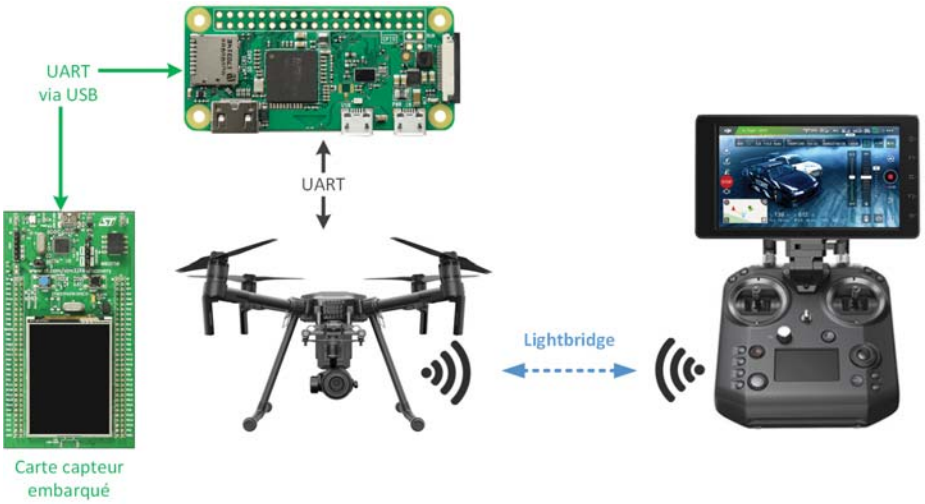


FIGURE 24: Schéma de principe général

Le moniteur Crystal Sky est directement relié à la télécommande Cendence via son socle par USB Type C. La télécommande et le drone communiquent via la connexion Lightbridge implémentée par DJI

Le Pi communique en UART via ses GPIO dédiées avec le Matrice 210. Il utilise le port `/dev/ttyAMA0` à une vitesse de 230'400 Bauds.

Une carte STM32 s'occupe du traitement de signal de l'antenne DVA et communique également en UART. Le Pi Zero ne disposant que d'un seul UART, un adaptateur USB-TTL est relié au port USB utilisateur. La communication fonctionne sur le port `/dev/ttyUSB0` à une vitesse de 115'200 Bauds.

Les communications UART fonctionnent en 3.3V TTL.

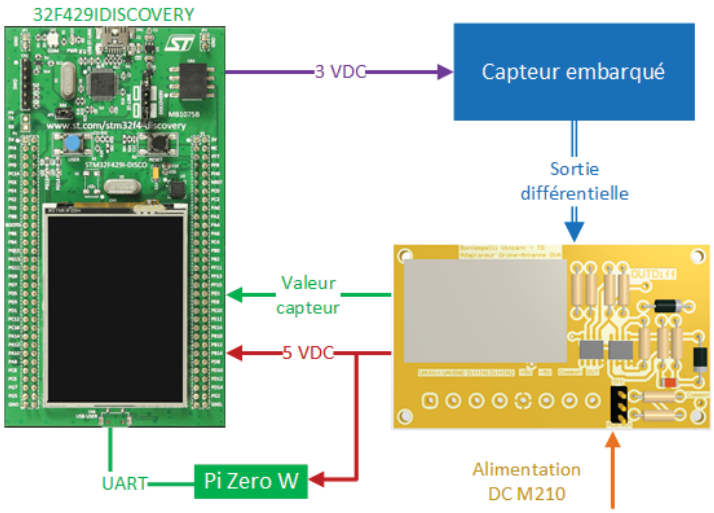


FIGURE 25: Schéma de principe - Capteur embarqué

Le Pi ainsi que la carte STM32 sont alimentés via une carte réalisée par Vincent Bontempelli dans le cadre de son travail de bachelor. Elle est alimentée par la sortie 18-26V du port d'extension du Matrice 210 et possède un DC/DC Traco Power afin de fournir une sortie ±5 VDC. Cette carte est également chargée de traiter la sortie fournie par le capteur

embarqué, ce dernier est présenté plus en détail dans le chapitre 12.

Un schéma de connexion complet et détaillé est disponible en annexe.

La structure du code développé repose sur un CMakeLists. Les manipulations concernant le déploiement et l'utilisation du logiciel sont disponibles dans le *README.md* du code sur son Github [1]. Actuellement, le programme se trouve dans le répertoire
`~/Matrice210/code/Matrice210/build/bin`
de la session `Pi` du Raspberry Pi Zero. Le mot de passe de la session est `M210PiZeroW`.

Un service Linux a été créé afin de lancer automatiquement le programme lorsque le Pi est mis sous tension. Si le programme s'arrête, il est également redémarré. Un système de logs a été établi en sauvegardant la console du logiciel dans un fichier numéroté et horodaté. A nouveau, plus d'informations sont disponibles dans le *README.md* du code.

11.3 Configuration de debug

Pendant les phases de développement, des connexions supplémentaires étaient nécessaires pour déboguer le code des différents périphériques.

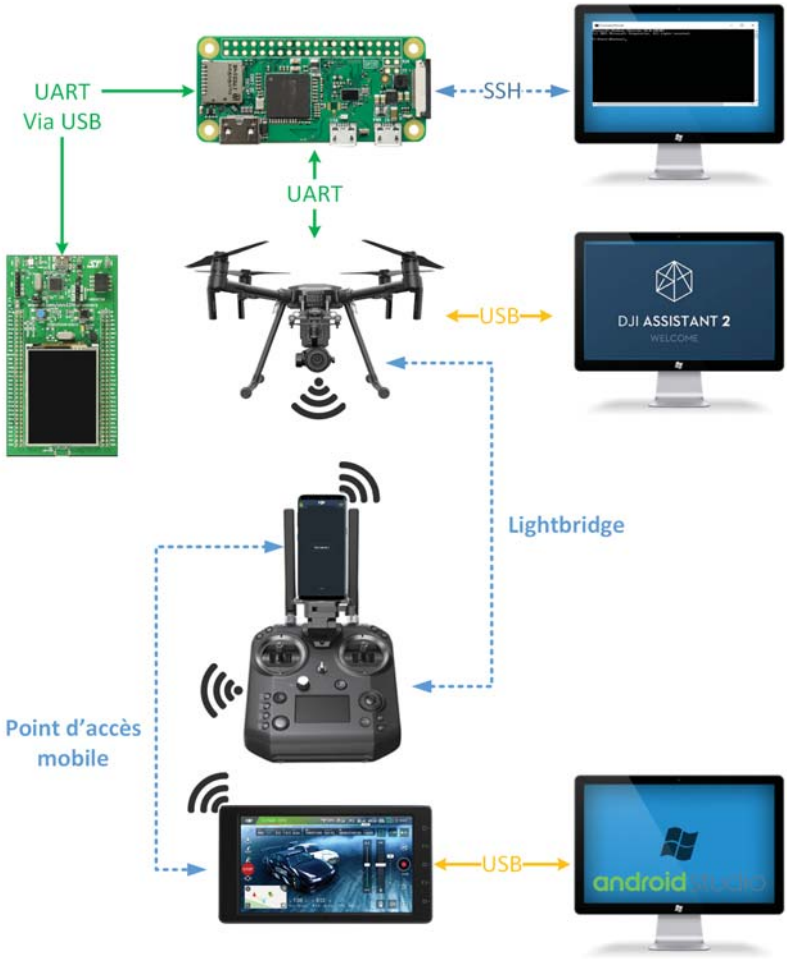


FIGURE 26: Configuration de debug

1. Solution sur Raspberry Pi : `https://github.com/jonathanmichel/Matrice210Pi`

Le code du système embarqué a été développé sous *CLion* et déployé en SSH sur le Raspberry Pi. Pour ce faire, il est nécessaire de connecter le Pi au réseau *device-hevs*. En effet, le réseau habituel *secure-hevs* ne permet pas d'accéder aux périphériques connectés à distance. De plus il ne s'agit pas du même réseau que celui utilisé par les postes fixes en salle de classe. Un ticket a été ouvert auprès du SInf pour autoriser l'adresse MAC du Pi Zero sur le réseau *device-hevs* et obtenir les informations d'authentification. Ce dernier est disponible en annexe.

A noter que le réseau *device-hevs* est parfois lent, il est plus agréable de brancher un écran et un clavier sur le Raspberry Pi Zero pour les manipulations.

Le drone est relié par USB à un ordinateur qui fait tourner le logiciel DJI Assistant présenté au chapitre 6.4.1 afin de configurer le Onboard SDK et d'utiliser le simulateur de vol.

La connexion entre la télécommande et le moniteur tactile est mise en place à l'aide de l'application DJI Bridge présentée au chapitre 6.4.4.

Le moniteur tactile est directement relié par USB à l'IDE *Android Studio* utilisé pour le développement de l'application Android.

11.4 Montage sur le drone

Le système embarqué et les cartes de traitement de signal sont fixés sur le drone grâce à des plaques d'acrylique conçues à la découpeuse laser. Le Matrice 210 dispose de trous de fixations sur le dessus. L'adaptation de l'antenne sous le drone a été réalisée par Vincent Bontempelli dans le cadre de son travail de bachelor.

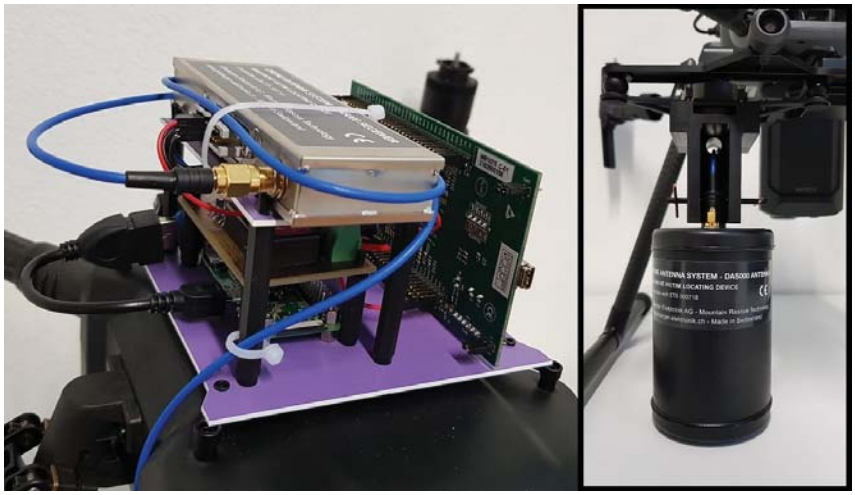


FIGURE 27: Montage du système embarqué et de l'antenne sur le drone

11.5 Architecture logicielle

Le logiciel embarqué a été séparé en plusieurs parties. Chacune de ces parties regroupées en package est présentée en détail dans les chapitres suivants.

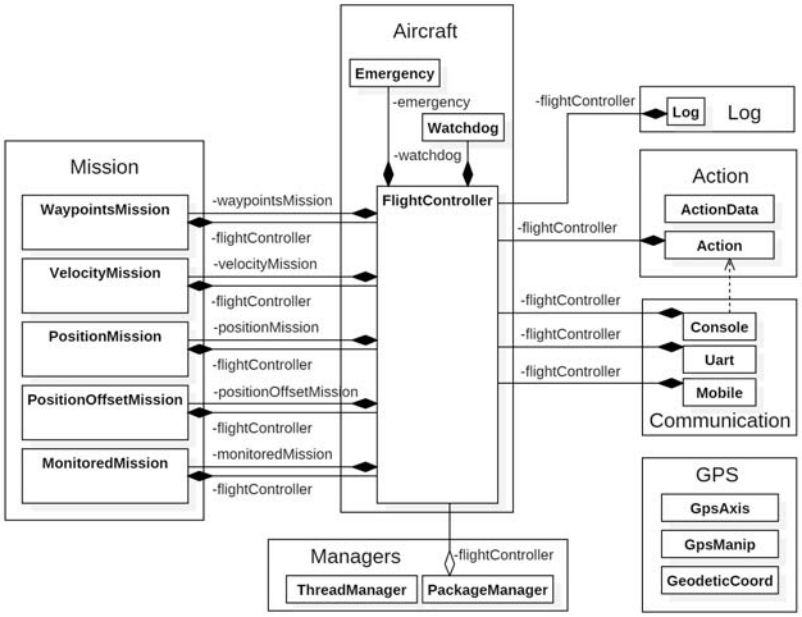


FIGURE 28: Architecture logicielle système embarqué

— **Aircraft** - Ce package contient les classes chargées de gérer le déroulement du vol. La classe `FlightController` est la seule à interagir avec le OSDK. Les classes `Emergency` et `Watchdog` fournissent des fonctionnalités liées à la sécurité.
— **Communication** - Ce package contient les classes chargées de gérer les différentes interfaces existantes pour interagir avec le déroulement du code. La classe `Console` permet à l'utilisateur de rentrer des commandes dans la console du programme, la classe `Mobile` gère la communication avec le MSDK et la classe `Uart` communique avec le capteur embarqué.
— **Action** - Ce package contient une classe `Action` qui met à disposition une queue destinée à stocker des `ActionData`, il s'agit d'objets crées dans le package `Communication` contenant des actions utilisateurs et leurs paramètres respectifs.
— **Managers** - Ce package contient un `ThreadManager` qui met à disposition des méthodes statiques pour la création de threads et un `PackageManager` chargé de gérer les informations de télémétrie reçues par le drone.
— **Missions** - Ce package propose différents types de missions exécutables par le drone.
— **Log** - La classe `Log` fournit des méthodes de debug via la console et la communication Mobile-Onboard.
— **GPS** - Ce package met à disposition des méthodes de calculs pour la localisation GPS dans la classe `GpsManip` en fonction de coordonnées de type `GeodecticCoord`. La classe `GpsAxis` permet de modifier le référentiel utilisé par le drone pour se déplacer.

11.6 Contrôleur de vol

Le contrôleur de vol est le principal élément du logiciel, il est chargé de gérer le bon déroulement du vol. Il gère l'état du drone et envoie en continu les ordres de déplacement au Onboard SDK lorsque le drone exécute une mission grâce à une machine d'état. Cette dernière est exécutée dans un thread dédié. Toutes les classes désirant interagir avec le drone passent par le contrôleur de vol.

Le contrôleur de vol est implémenté dans le package `Aircraft`.

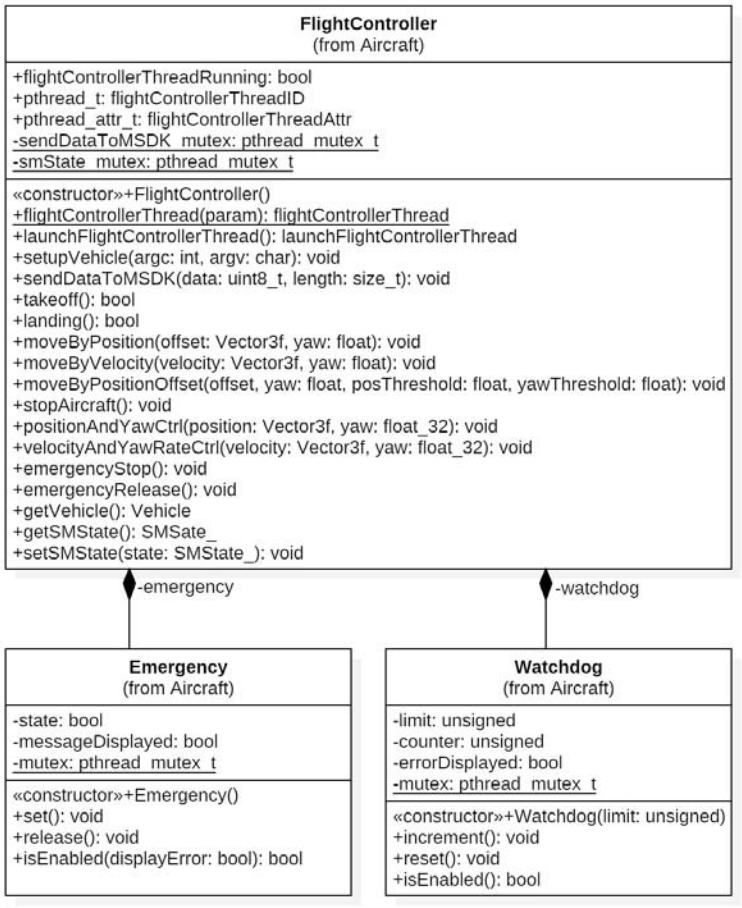


FIGURE 29: Package Aircraft

FlightController
Le contrôleur de vol est implémenté dans la classe `FlightController`.

Watchdog
Un watchdog a été implémenté afin de garantir que la connexion avec la télécommande et en particulier l'application implémentant le MSDK soit maintenue. Ceci pour des raisons de sécurité. Ainsi si l'utilisateur demande au drone de se déplacer à une vitesse constante et que la connexion est perdue, le déplacement sera arrêté et le drone se stabilisera en vol. Ce watchdog est géré par la classe du même nom.

Emergency
La classe `Emergency` gère l'état d'arrêt d'urgence du drone.

11.6.1 Control Authority

Étant donné qu'il y a plusieurs sources de contrôle pour le drone, le contrôleur de vol du Matrice 210 choisit laquelle a la priorité en tout temps. Les différentes sources de contrôle triées par priorité de la plus haute à la plus faible sont :

1. La télécommande
2. L'application mobile basée sur le MSDK connectée à la télécommande
3. L'ordinateur de bord basé sur le OSDK connecté au drone

Télécommande
La télécommande a la plus haute priorité de contrôle et peut récupérer ce dernier en tout temps. Pour ce faire, il est nécessaire de changer de mode de vol sur la télécommande comme vu au chapitre 6.2.

Mobile SDK
Dans le cas de ce travail de bachelor, le MSDK ne transmet pas d'ordres de déplacement au drone. Les commandes envoyées depuis l'application sont reçues par l'ordinateur de bord qui lui va communiquer avec l'aéronef.

Onboard SDK
Pour que le OSDK puisse piloter le drone, plusieurs conditions doivent être respectées :

— Le mode de vol de la télécommande doit être défini sur la position P
— L'ordinateur de bord doit avoir activé le Onboard SDK
— L'ordinateur de bord doit avoir demandé et obtenu le contrôle du drone

Si ces conditions ne sont pas respectées, l'ordinateur de bord ne pourra pas déplacer le drone mais recevra tout de même les informations des capteurs ainsi que celles du statut et pourra exécuter des actions de base tel que le décollage et l'atterrissage.

La figure 30 présente les fonctionnalités accessibles par l'ordinateur de bord et la télécommande pendant les différentes étapes du déroulement de l'application.

On y constate un cas qui mérite une attention particulière. Celui-ci a lieu lorsque le Onboard SDK est activé, que le contrôle du drone a été obtenu et que des ordres de déplacement ont été transmis. Si l'utilisateur modifie le switch de mode de vol alors que les multiples modes ne sont pas activés, le drone revient dans l'état précédent et la télécommande reprend le contrôle des déplacements. Mais étant donné que les ordres sont envoyés en continu par l'ordinateur de bord (voir chapitre 11.10.2), le drone va instantanément se remettre à bouger. Si au contraire les multiples modes de vol sont activés, il faudra repasser le drone en mode P et l'ordinateur de bord devra redemander le contrôle du drone.

Pour cette raison il est conseillé à l'utilisateur d'activer ce mode. La démarche se trouve au chapitre 6.2. L'évitement d'obstacle est également garanti par le drone lorsqu'il fonctionne en mode P et qu'il exécute des ordres transmis par le Onboard SDK.

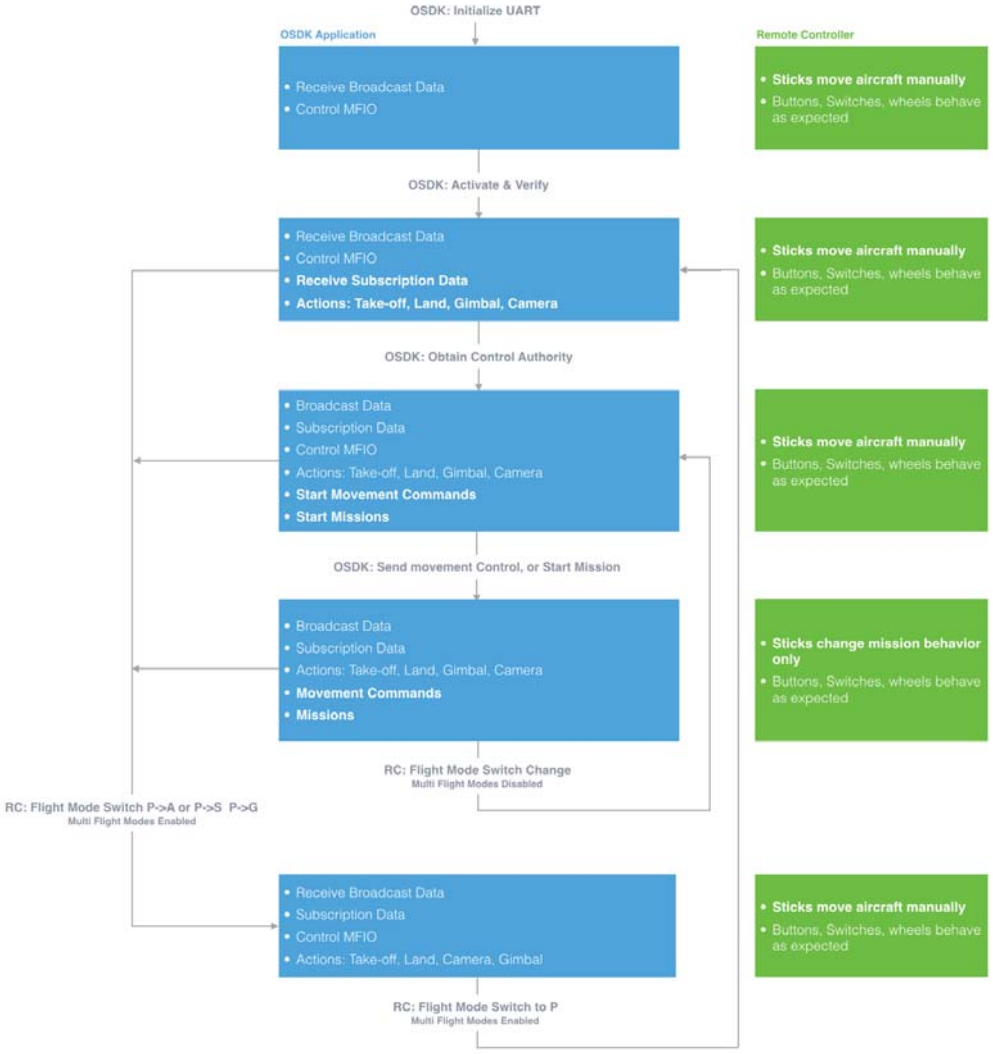


FIGURE 30: Control Authority [1]

11.6.2 Activation du Onboard SDK

Comme vu précédemment, le Onboard SDK doit être activé par l'ordinateur de bord pour être utilisé. Pour ce faire, DJI demande au développeur de s'enregistrer sur leur site et de générer une clé et un id qui seront transmis au drone en début de programme [2].

1. Source : `https://developer.dji.com/onboard-sdk/documentation/guides/component-guide-control-authority.html`
2. Enregistrement sur le site de DJI : `https://developer.dji.com/register/t`

11.7 Communication

Il existe plusieurs interfaces pour interagir avec le déroulement du code : la console du logiciel sur le Pi, la connexion UART avec la carte STM32 du capteur embarqué et la communication Mobile avec l'application Android au sol. Le package `Communication` contient les classes chargées de gérer ces différentes interfaces.

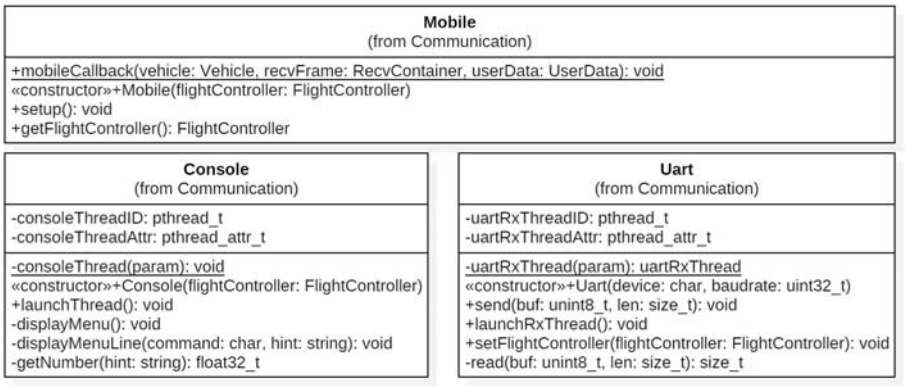


FIGURE 31: Package Communication

Console
La classe `Console` démarre un thread dédié à la réception et au traitement des commandes entrées dans la console du logiciel.

Uart
Comme expliqué dans le chapitre 11.2, le capteur embarqué communique en UART avec le Raspberry Pi. La classe `Uart` démarre un thread chargé de lire en continu les données reçues sur le port UART dédié à cette connexion. Un protocole simple a été implémenté et gère uniquement la réception de données depuis la carte STM32. Les valeurs ASCII des nombres à transmettre sont envoyés, le caractère '|' fait office de séparation entre les différentes valeurs et le caractère '@' indique la fin d'une trame.

Ainsi lorsque la carte STM32 transmet 0|4123@ le logiciel sait que le capteur (0) vaut actuellement 4123. Cette valeur est directement retransmise au MSDK pour être affichée sur l'application Android.

Au besoin d'autres commandes peuvent être codées, comme 1|0|-5|3|0|90@ qui pourrait permettre de déplacer le drone et signifier :

— 1 : Velocity mission
— 0 : Démarrer
— -5 : Vélocité en X en mètres par seconde
— 3 : Vélocité en Y en mètres par seconde
— 0 : Vélocité en Z en mètres par seconde
— 90 : Angle du yaw en degrés

A noter que l'implémentation actuelle permet également d'envoyer des valeurs décimales.

Mobile
La classe `Mobile` configure à l'aide d'une méthode proposée par le OSDK la méthode de callback à utiliser lors d'une réception de données depuis le MSDK. Cette méthode va décoder la trame reçue et ajouter à la queue d'action les actions à exécuter et leurs paramètres. A noter que pour des raisons de sécurité évidentes l'arrêt d'urgence est directement exécuté et non ajouté à la queue d'actions.

11.8 Actions

Comme vu précédemment, l'exécution du code peut être influencée de plusieurs façons. Afin de stocker les ordres reçus pour les traiter dans un second temps, un système de queue a été mis en place. Ceci afin de minimiser le temps passé dans les différentes méthodes de réception de données afin de garantir qu'aucune information n'est perdue. Ceci permet également de traiter le plus rapidement possible un arrêt d'urgence par exemple, qui lui ne sera pas ajouté à la queue d'actions mais directement exécuté afin de garantir la sécurité du système. La gestion de cette queue se fait dans la classe `Action`, elle est continuellement traitée dans le main.

ActionData

Afin de pouvoir ajouter des actions avec des paramètres dont la taille n'est pas toujours identique un container `ActionData` a été créé. Il s'agit d'un objet réservant une taille définie à l'instanciation en mémoire dynamique. Des données de différents types peuvent être ajoutées puis récupérées. Actuellement tous les types ne sont pas supportés. La classe se charge d'indiquer à l'utilisateur lorsque l'intégralité de la mémoire allouée est remplie ou lorsqu'au contraire elle est vide et qu'il tente de récupérer une valeur.

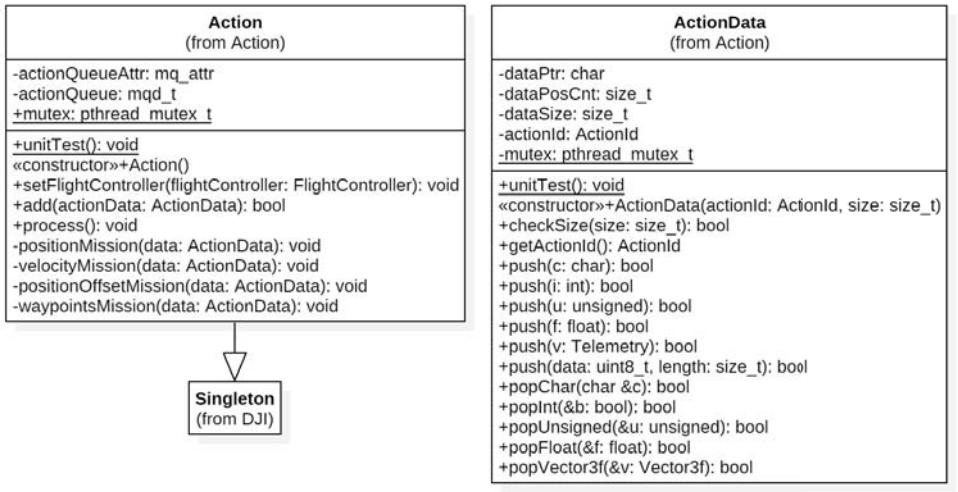


FIGURE 32: Package Action

11.9 Télémétrie

La télémétrie est la principale source de mesures en temps réel et de transmission d'informations de statut. Le OSDK fournit des APIs pour lire les données de télémétrie de deux façons : en broadcast ou en subscription.

11.9.1 Broadcast

En mode broadcast, les paquets de données arrivent à une fréquence préconfigurée. Celle-ci peut être changée avant le vol avec le logiciel DJI Assistant 2 ou pendant le vol grâce au OSDK.

Les données arrivent en package contenant les informations des composants ci-dessous :
— Timestamp
— Hardware Synchronization Timestamp
— Quaternion
— Angular Rate
— Velocity
— Gyroscope Reading
— Global Position
— Relative Position
— GPS
— RTK
— Magnetometer
— Remote Controller
— Gimbal
— Flight Status
— Battery

Le broadcast est disponible à 0, 1, 10, 50 et 100Hz. La fréquence peut être définie pour chaque composant individuellement. En pratique, si le développeur souhaite recevoir la vélocité à 50 Hz et le niveau de batterie à 10Hz, un paquet global sera transmis à 50 Hz par le broadcast mais les informations concernant la batterie n'arriveront qu'une fois sur cinq.

11.9.2 Subscription

La télémétrie par Subscription a été introduite par DJI depuis le Onboard SDK 3.3. Elle a la même utilité que le broadcast mais propose une plus grande variété d'informations récupérables et une programmation plus flexible pour les fréquences d'émissions.

L'utilisateur peut choisir des `Topics` et les ajouter à un package de Subscription qui sera configuré pour arriver à une fréquence définie. Le Onboard SDK met à disposition cinq packages d'une taille maximum de 300 bytes, ce qui permet à l'utilisateur d'ajouter autant de `Topics` que désiré. La fréquence d'émission peut être configurée individuellement pour chaque package.

Les composants ci-dessous peuvent être ajoutés à un package tant que leur fréquence maximale est plus élevée que celle désirée par l'utilisateur :
— Hardware Synchronization : 400Hz
— Quaternion : 200Hz
— Acceleration : 400Hz
— Angular Rate : 400Hz
— Velocity : 200Hz
— Barometer Altitude : 200Hz
— GPS : 50 Hz
— RTK : 50 Hz

— Compass : 100 Hz
— Remote Controller : 50 Hz
— Gimbal : 50 Hz
— Flight Status : 50 Hz
— Battery : 50 Hz
— Display Mode : 50 Hz
— Landing Gear : 50 Hz
— Motor : 50 Hz

Le broadcast et les subscriptions peuvent être utilisés simultanément.

PackageManager

La classe `PackageManager` a pour objectif de fournir une interface facile à utiliser pour la gestion des packages. Elle permet à l'utilisateur de ne plus avoir à se soucier du nombre de package déjà en cours d'utilisation et s'assure du bon déroulement lors de la création d'un nouveau package.

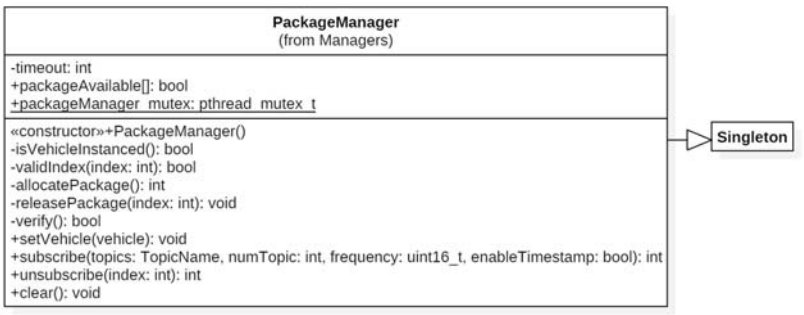


FIGURE 33: PackageManager

11.10 Missions

11.10.1 Systèmes de coordonnées

Avant de présenter les différents types de missions possibles, ils convient de faire un rappel sur le lexique utilisé en aéronautique pour le contrôle de vol. La description des mouvements d'un aéronef dépend de sa position et de l'orientation des axes qui déterminent un système de coordonnées. Il en existe plusieurs mais les deux utilisés par le Onboard SDK sont relatifs au fuselage du drone (body frame) ou relatifs au sol (ground/world frame).

Body frame
Ce système de coordonnées est relatif au fuselage du drone. Trois axes perpendiculaires sont définis en partant du centre de masse de ce dernier. L'axe **X** pointe vers l'avant de l'appareil, l'axe **Y** vers la droite, et, en suivant la règle de la main droite, l'axe **Z** pointe vers le bas. Les translations en positif sur les axes X, Y et Z sont respectivement définies par des déplacements en avant, vers la droite et en bas.

La rotation est également décrite avec ces mêmes axes en utilisant la règle de la main droite pour définir la direction positive de rotation. Lorsque l'on parle de rotation, les axes X, Y et Z sont respectivement nommés **Roll**, **Pitch** et **Yaw**.

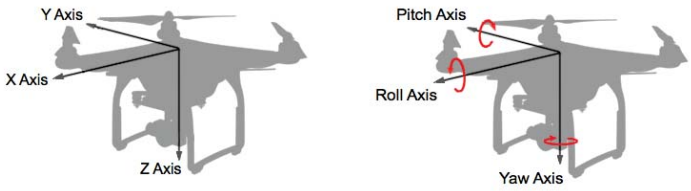


FIGURE 34: Body frame et rotation [1]

Ground frame
Un autre système de coordonnée très utilisé est le ground frame (ou world frame). Il consiste à aligner les axes X et Y avec les directions nord et est. En suivant la règle de la main droite, l'axe Z pointe vers le bas. Ainsi un angle de 0° autour de celui-ci pointe vers le Nord et un angle de +90° vers l'Est. Cette convention s'appelle le North-East-Down (NED).

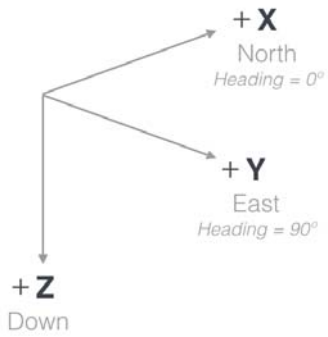


FIGURE 35: Ground frame [2]

L'origine du système de coordonnées NED est un point fixe comme la position de décollage du drone.

1. Source : https://developer.dji.com/mobile-sdk/documentation/introduction/flightController_concepts.html
2. cf. 1

11.10.2 Pilotage

Il existe plusieurs possibilités fournies par DJI pour piloter le drone avec le Onboard SDK. Tous utilisent le système de coordonnées NED. Quatre modes de contrôle sont disponibles :

— Velocity Control - Contrôler la vélocité du drone en X, Y, et Z et la vitesse angulaire du yaw.
— Position Control - Contrôler la position du drone en X, Y et Z ainsi que l'angle du yaw.
— Attitude Control - Contrôler l'attitude du drone ainsi que l'altitude en Z. L'attitude est l'orientation du drone définie par la rotation autour du roll, du pitch et du yaw.
— Angular rate Control - Contrôle le taux d'attitude ainsi que l'altitude du drone.

Lors de l'utilisation de ces modes, il est nécessaire d'envoyer en continu à 50 Hz les ordres de déplacement au drone. Si un ordre n'est plus transmis, le drone s'arrête et se stabilise en vol. Cet envoi continu est géré par la machine d'état du thread dédié de la classe `FlightController` comme expliqué au chapitre 11.6.

Une dernière possibilité de pilotage est l'utilisation de Waypoints Missions qui permettent de définir des points GPS que le drone doit atteindre.

Ces différentes possibilités ont été utilisées afin de proposer plusieurs types de missions pour déplacer le drone. Certaines sont inspirées des codes exemples fournis par DJI sur le Github du Onboard SDK [1]. Le contrôle par attitude et par taux d'attitude n'est pour l'instant pas implémenté.

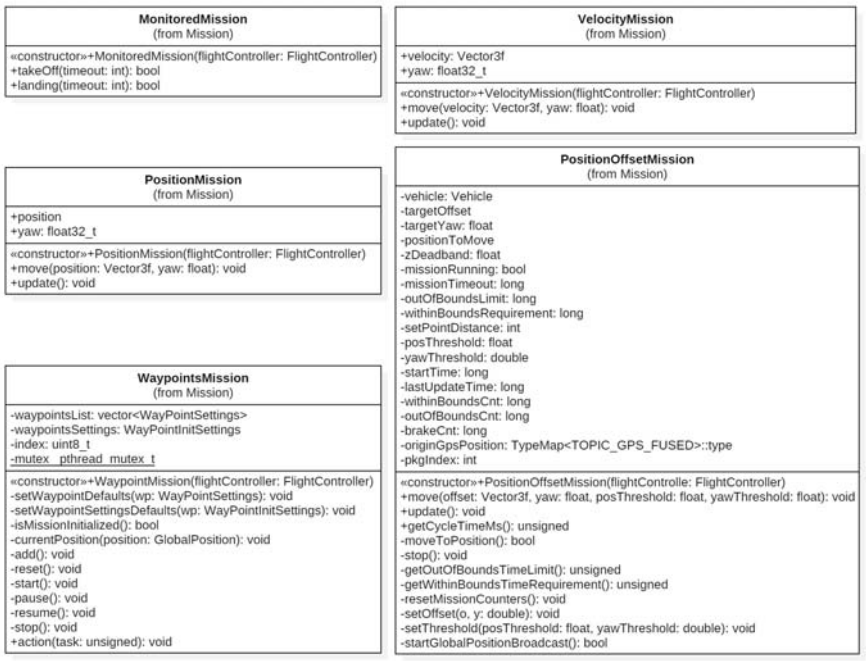


FIGURE 36: Package Missions

1. Code exemple Onboard SKD `https://github.com/dji-sdk/Onboard-SDK/tree/3.7/sample/linux`

Monitored missions
La classe `MonitoredMissions` met à disposition deux méthodes bloquantes pour faire décoller et atterrir le drone. Ces dernières se chargent de vérifier le bon déroulement des opérations tout au long de celles-ci.

Velocity mission
Il est possible de déplacer le drone en lui donnant un vecteur de vélocité en mètres par seconde en X, Y et Z selon le référentiel NED ainsi qu'une vitesse angulaire en degrés par seconde sur son yaw. La classe `VelocityMission` implémente cette possibilité.

Position mission
Le mode `Position` permet de définir une altitude absolue en mètres relative à celle de la position de décollage ainsi qu'un angle de yaw absolu en degrés à atteindre. Les valeurs X et Y transmises dans ce mode indiquent des positions relatives à atteindre. La classe `PositionMission` implémente ce type de contrôle.

Position offset mission
Étant donné que le mode `Position` présenté ci-dessus demande des valeurs relatives en X et Y mais des valeurs absolues en Z et yaw, une solution a été développée afin de pouvoir donner au drone un offset de position à atteindre. La classe `PositionOffsetMission` se charge ainsi d'envoyer en continu les ordres de déplacement adéquats en fonction de la position actuelle du drone et de l'offset désiré.

Waypoints mission
Les `Waypoints missions` sont proposées par DJI. Il s'agit d'une série d'emplacement que le drone va atteindre. Un emplacement est composé d'une latitude, d'une longitude et d'une altitude. L'implémentation dans la classe `WaypointsMission` permet à l'utilisateur d'ajouter la position actuelle du drone à l'itinéraire puis de rejouer la séquence d'emplacement dans l'ordre ou elle a été ajoutée. Pendant le déroulement, la mission peut être mise en pause, relancée ou arrêtée.

Avalanche mission
Les `Avalanche missions` sont encore en cours de développement. Elles rentrent dans le cadre de l'utilisation du drone pour la recherche de victimes d'avalanches. Étant donné qu'elles ne concernent pas le cahier des charges de ce travail de bachelor et en vue de l'avancement de celles-ci, elles ne seront pas plus détaillées ici.

11.11 Positionnement GPS

11.11.1 Système géodésique

Afin de déterminer la position du drone, la télémétrie renvoie les coordonnées géodésiques de cette dernière. Un système géodésique repose sur un ellipsoïde de révolution [1] qui a pour but de représenter le géoïde [2] de la terre. Un point P y est localisé par des coordonnées exprimées en valeurs angulaires par la longitude λ, la latitude Φ et la hauteur géodésique h.

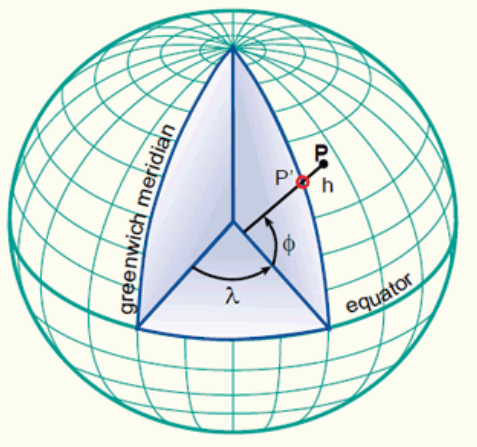


FIGURE 37: Point P dans un système géodésique [3]

La longitude mesure l'angle entre le premier méridien et le point mesuré. Elle varie de -180° à +180°. La latitude mesure l'angle que fait la normale à l'ellipsoïde de référence en ce point avec le plan équatorial. Elle varie de -90° à +90°. Finalement la hauteur est mesurée le long de la normale à l'ellipsoïde, h est petit à proximité de la surface terrestre.

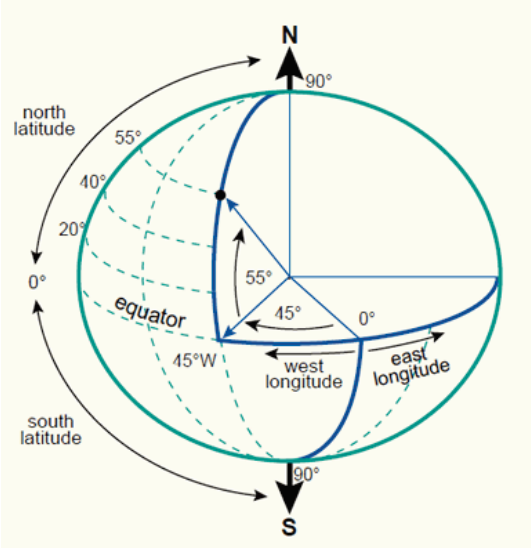


FIGURE 38: Longitude et latitude [4]

1. Surface de révolution obtenue par rotation dans l'espace d'une ellipse autour de l'un de ses axes
2. Surface théorique de la pesanteur, qui correspond au niveau moyen des mers
3. Source : `http://kartoweb.itc.nl/geometrics/coordinate%20systems/body.htm`
4. cf. 3

A noter qu'il existe plusieurs types de latitude, le Onboard SDK retourne la latitude géodésique comme expliqué ci-dessus. Dans le cas des calculs et afin de les simplifier, cette latitude sera considérée comme étant géocentrique. En pratique elles ne diffèrent que de quelques minutes et vu les distances en jeux cela n'a pas d'influence.

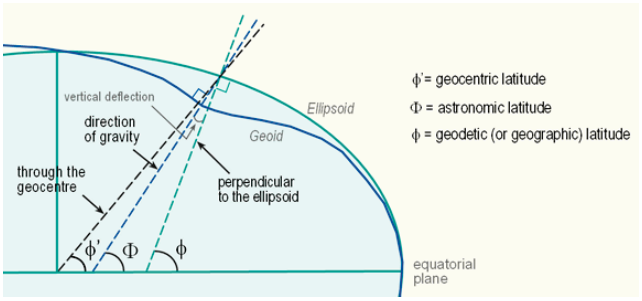


FIGURE 39: Différentes latitudes existantes [1]

La classe `GeodeticCoord` permet de stocker une position géodésique avec sa latitude et sa longitude en radian et fournit les interfaces nécessaires pour récupérer les coordonnées en degré ou en DMS (Degré Minute Secondes). Actuellement l'altitude n'est pas sauvegardée dans cette classe car elle n'est pas nécessaire.

11.11.2 Calculs de position

La classe `GpsManip` met à disposition une méthode permettant de convertir deux positions géodésiques en un offset NED. Pour rappel, la distance X de cet offset représente le déplacement vers le Nord et la distance Y vers l'Est.

Pour ce faire, elle considère la latitude fournie comme étant géocentrique. L'offset NED $N(x, y)$ entre deux coordonnées O (Origin) et T (Target) respectivement de latitude Φ_O et Φ_T et de longitude λ_O et λ_T se calcule principalement avec la formule de longueur d'arc de cercle :

$$\Delta_\Phi = \Phi_T - \Phi_O$$
$$\Delta_\lambda = \lambda_T - \lambda_O$$
$$N_x = \Delta\Phi \cdot R_{earth}$$

La latitude moyenne entre les deux coordonnées est calculée. Ceci est nécessaire car la latitude aura une influence sur la distance Est-Ouest que représentera un delta en longitude.

$$\Phi_{AVG} = \frac{\Phi_O + \Phi_T}{2}$$

Le rayon d'une coupe de la terre parallèle au plan équatorial à cette latitude est calculé. La terre est considérée comme parfaitement sphérique.

$$R_{cut} = R_{earth} \cdot cos\Phi_{AVG}$$
$$N_y = \Delta_\lambda \cdot R_{cut}$$
$$N(N_x, N_y)$$

Le rayon équatorial de la terre est utilisé, il vaut 6378137 mètres.

La classe `GeodeticCoord` dispose de surcharges d'opérateurs permettant d'ajouter un Vecteur NED 2D à une coordonnée géodésique. Le fonctionnement est similaire au développement présenté ci-dessus.

1. cf. 3

En plus de cette méthode, la classe `GpsManip` contient d'autres méthodes de calcul vectoriel afin de, par exemple, mesurer l'angle entre deux vecteurs NED ou encore faire des rotations vectorielles. A nouveau l'implémentation actuelle prend en compte uniquement deux dimensions.

Grâce à ces méthodes, la classe `GpsAxis` permet à l'utilisateur de modifier l'orientation du plan X-Y pour que l'axe X de celui-ci ne pointe plus vers le Nord. Elle s'occupe de projeter les coordonnées dans le référentiel désiré par l'utilisateur en coordonnées NED utilisables par le Onboard SDK.

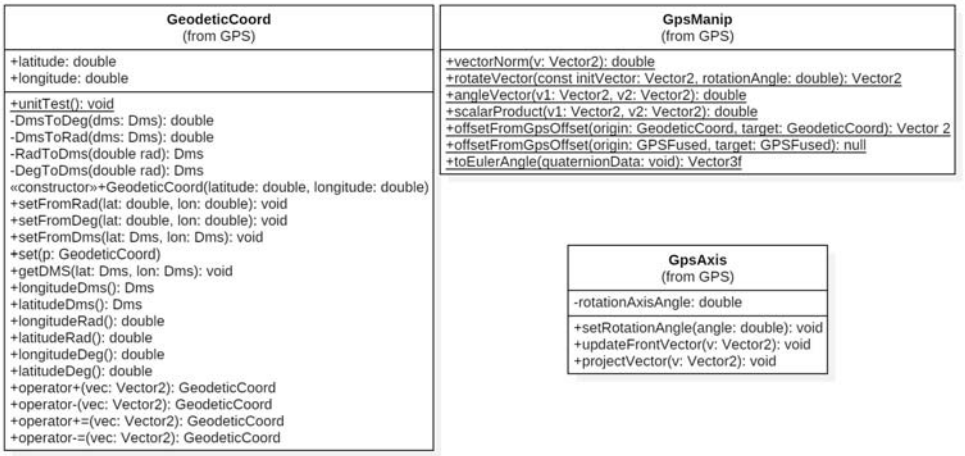


FIGURE 40: Package GPS

Les manipulations sur les coordonnées GPS présentées ci-dessus ont été testées séparément sur Qt avant d'être implémentées dans le logiciel sur Raspberry Pi. Plus d'informations sont disponibles au chapitre 17.

12 Capteur DVA embarqué

Durant le travail de bachelor, il a été convenu que le capteur embarqué serait l'antenne DVA utilisée par Vincent Bontempelli pour son TB.

12.1 Technologie DVA

Un DVA (Détecteur de Victimes d'Avalanche) est un appareil électronique émetteur-récepteur d'un signal radio à 457kHz. Il permet de localiser les victimes d'avalanches enfouies sous la neige si ces dernières en sont également équipées.

FIGURE 41: DVA Barryvox de Mammut [1]

12.2 Antenne DVA

Girsberger est une société suisse spécialiste mondial dans le sauvetage avalanche. Elle a récemment développé un dispositif adapté à la recherche DVA par drone, le *DAS000*. Le produit est composé d'une antenne et d'un récepteur qui fournit une sortie différentielle représentant le champ magnétique mesuré. Lorsque ce dernier augmente, l'intensité du signal fait de même. La sensibilité de l'antenne peut être modifiée sur neuf niveaux grâce au récepteur, cette fonctionnalité n'est pas implémentée pour l'instant. Elle est définie en hardware sur la sensibilité minimale.

FIGURE 42: DAS000 de Girsberger [2]

1. Source : https://www.boosport.ch/42707-thickbox/dva-arva-barryvox-s-mammut.jpg
2. Source : https://www.girsberger-elektronik.ch/das000-drone-antenna-system

12.3 Traitement de signal

Une carte électronique a été développée par Vincent Bontempelli dans le cadre de son travail de bachelor. Elle traite le signal différentiel transmis par le récepteur et fournit une sortie analogique 0-4V de l'enveloppe de ce dernier. Le Pi ne disposant pas d'entrée analogique, il est nécessaire de récupérer l'information différemment. Deux solutions ont été envisagées : les entrées analogiques du port d'extension du Matrice 210 ou une carte externe communiquant avec le Pi. Mon collègue a choisi de faire des tests d'implémentation d'algorithme sur une carte 32F429IDISCOVERY. Dans un premier lieu, nous avons ainsi décidé d'embarquer cette carte sur le drone.

Le signal DVA est pulsé à 1Hz pour une durée de pulse d'environ 80 à 100ms. Afin de transmettre au drone une valeur utilisable d'intensité de champ magnétique, le signal est traité par la carte STM32. Quarante mesures sont faites par secondes et la valeur maximale est transmise à l'aide du protocole présenté au chapitre 11.7. Il s'agit d'un traitement de signal sommaire qui mériterait d'être amélioré pour une utilisation plus poussée mais qui convient parfaitement dans le cadre de ce travail de bachelor. Le code déployé sur la carte STM32 est disponible sur Github [1]. A noter que cette partie a été réalisée par mes soins.

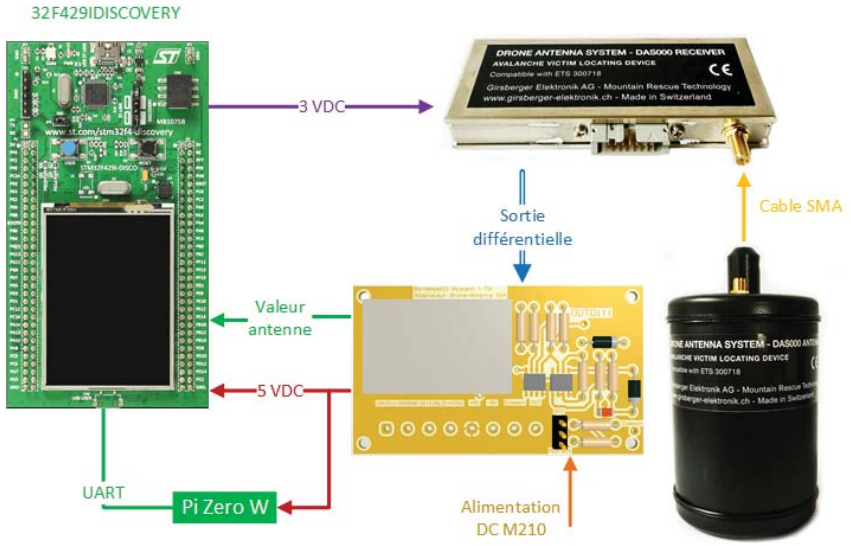


FIGURE 43: Schéma de principe - Capteur embarqué

Le récepteur de l'antenne est alimenté par la sortie 3VDC de la carte STM32. Un schéma de connexion complet et détaillé est disponible en annexe.

1. Code déployé sur STM32 : `https://github.com/jonathanmichel/Matrice210Stm32/`

13 Application Android

L'application a été développée sur *Android Studio 3.1.4*. Le code est disponible sur Github [1].

Le code exemple mis à disposition par DJI pour le UXSDK [2] a permis de mettre en place la structure du projet Android et d'y ajouter les dépendances nécessaires pour obtenir le MSDK. Tout comme pour le Onboard SDK, il est nécessaire de s'inscrire sur le site de DJI pour générer une clé afin d'activer le Mobile SDK [3].

L'application se sert de toasts pour afficher des informations sur l'écran [4]. Pour que ceux-ci apparaissent, il est nécessaire que cette dernière ait la permission d'afficher des notifications dans les paramètres Android.

13.1 Architecture logicielle

L'application est composée d'une seule activité, celle-ci est chargée de gérer la connexion avec la télécommande, de transmettre les données au Onboard SDK et de rediriger les données reçues pour qu'elles soient correctement affichées.

Elle affiche une barre de statut contenant toutes les informations de connexion similaire à celle proposée sur l'application officielle DJI Go 4. Comme expliqué au chapitre 8.4, il s'agit de widgets mis à disposition par le User eXperience SDK de DJI.

Le statut à gauche indique lorsque le drone est connecté à la télécommande, il ne fonctionne pas toujours correctement. Il vaut mieux se référer aux niveaux de batteries et à l'icône de statut rond pour déterminer si le drone est connecté ou non. Le message juste à leur droite indique lorsque l'ordinateur de bord est connecté, ce statut est garanti par le watchdog présenté au chapitre 11.6. Finalement, un bouton permet de déclencher l'arrêt d'urgence pour arrêter tout déplacement du drone et le stabiliser en vol. Ce dernier est également déclenché quand l'application est arrêtée, mise en pause et restaurée.

FIGURE 44: Application Android - Barre de statut

L'espace en dessous de la barre de statut est réservé pour différents fragments, trois sont disponibles sur l'application actuellement et un quatrième a été développé pour tester la MOC mais n'est plus utilisé.

— **Dashboard Fragment** : C'est la vue affichée au démarrage de l'application, elle permet à l'utilisateur de choisir quelle interface il souhaite utiliser.
— **Pilot Fragment** : Cette interface est à utiliser pour piloter le drone en manuel.
— **Mission Fragment** : Cette interface permet de tester les différentes missions pour piloter le drone en autonome.
— **Mobile Fragment** : Cette interface servait pour faire les tests avec la communication Mobile-Onboard.

1. Code Android : `https://github.com/jonathanmichel/Matrice210AndroidApp`
2. DJI UXSDK pour Android : `https://github.com/dji-sdk/Mobile-UXSDK-Android`
3. Activation du Mobile SDK : `https://developer.dji.com/mobile-sdk/documentation/android-tutorials/ActivationAndBinding.html`
4. Toasts : `https://developer.android.com/guide/topics/ui/notifiers/toasts`

13.2 Dashboard fragment

Ce fragment s'affiche au démarrage de l'application, il permet à l'utilisateur de choisir l'interface qu'il souhaite utiliser.

Le champ de texte en fond de page permet de définir une adresse IP pour activer le mode IP Bridge présenté au chapitre 15. Lorsque l'utilisateur a fini d'entrer l'adresse du périphérique à utiliser, un appui sur la touche *Enter* démarre la connexion. DJI retourne un message "BridgeMode ON !" même si le second périphérique n'est pas connecté, ce message indique uniquement la configuration effective du mode. Pour plus d'informations, il faut se référer à l'application DJI Bridge tournant sur le périphérique connecté à la télécommande. La marche à suivre est disponible sur la documentation de DJI [1].

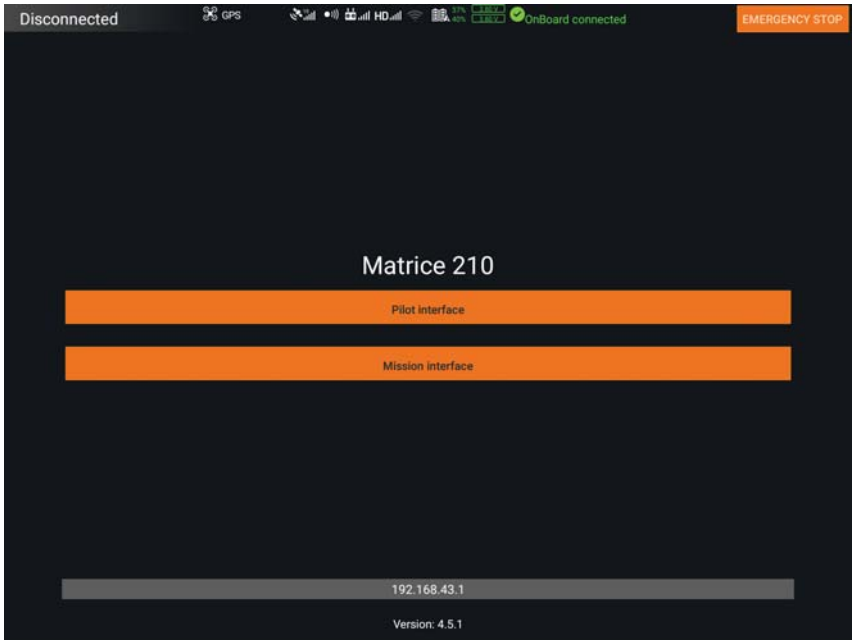


FIGURE 45: Application Android - Dashboard

1. Documentation DJI Bridge sur Android : `https://github.com/dji-sdk/Android-Bridge-App`

13.3 Pilot fragment

L'interface proposée par le `PilotFragment` est similaire à celle de l'application officielle DJI Go 4. Elle permet à un utilisateur de piloter le drone en manuel. Elle fournit un retour vidéo de la caméra et affiche des informations sur la vélocité et la position du drone ainsi qu'une carte qui peut être agrandie lorsque l'utilisateur clique dans le coin inférieur droit.

Deux widgets sont disponibles sur la gauche de l'écran pour décoller et atterrir automatiquement.

Les widgets sur la droite permettent de prendre des photos ou vidéos et de modifier les paramètres de la caméra. Les médias sont stockés sur une carte SD dans le drone. Ils peuvent être récupérés grâce à l'application DJI ou directement sur la carte.

FIGURE 46: Application Android - Interface Pilot

13.4 Mission fragment

L'interface proposée par le `MissionFragment` permet de tester les missions autonomes de pilotage du drone.

Les boutons *Take-off* et Landing démarrent les `MonitoredMission` présentées au chapitre 11.10.2. Contrairement aux widgets de décollage et d'atterrissage disponibles dans le `PilotFragment`, ces missions sont réalisées sur le système embarqué. Les widgets sont fonctionnels même si ce dernier ne répond plus, il s'agit de fonctionnalités proposées par DJI.

Une `PositionOffsetMission` ou une `VelocityMission` peut être exécutée. Les champs de texte *X*, *Y*, *Z* et *Yaw* permettent de définir les valeurs à utiliser. Elles seront respectivement en mètres et en degrés ou en mètres par seconde et en degrés par seconde en fonction du type de mission.

Pour les `WaypointsMission` le bouton *Add position* ajoute la position actuelle du drone à la liste des points à atteindre, cette liste peut ensuite être rejouée ou effacée. Pendant le déroulement de la mission, le déplacement peut être mis en pause, relancé ou arrêté.

La valeur du capteur embarqué est affichée, lorsqu'une nouvelle valeur est reçue le loader en début de ligne se met à jour.

Un bouton permet de récupérer le contrôle du drone via le OBSDK si l'utilisateur a changé de mode de vol sur la télécommande.

Finalement, un bouton permet de relâcher l'arrêt d'urgence.

FIGURE 47: Application Android - Interface Mission

13.5 Mobile fragment

L'interface du `MobileFragment` a été utilisée pour tester la communication Mobile-Onboard. Plus de détails sur les tests sont disponibles au chapitre 15. Elle n'est plus disponible sur la version actuelle de l'application. Au besoin, un tag `MOC_SpeedTest` est disponible sur le Github du code.

Elle permet d'envoyer une commande personnalisée au Onboard SDK.

Un test `Up` peut être démarré, les deux champs de texte disponibles servent respectivement à définir la taille en bytes de la trame à envoyer et le délai de transmission entre chaque trame en millisecondes.

L'utilisateur peut finalement démarrer un test `Ack-up` ou remettre à zéro les tests si un d'eux a échoué.

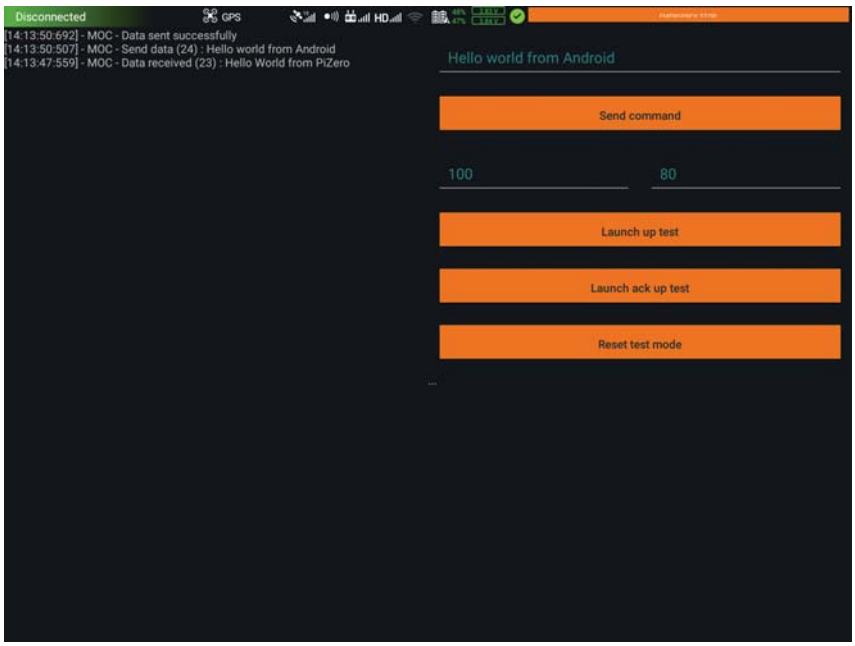


FIGURE 48: Application Android - Interface Mobile

Quatrième partie
Tests et résultats

14 Tests unitaires

Une partie des mécanismes implémentés dans le code du système embarqué disposent de tests unitaires, ces derniers sont exécutés en début de programme. C'est le cas des fonctionnalités liées aux actions et aux calculs de coordonnées. Lorsqu'ils se soldent par un échec le programme est interrompu. Ceci permet de garantir que le programme ne se lance pas avec des parties dysfonctionnelles.

15 Mobile-Onboard Communication

15.1 Débit

A l'aide de l'interface Mobile présentée au chapitre 13.5, le débit de la communication Mobile-Onboard a été estimé. Le code utilisé pour ces tests est disponible sur les Github du Raspberry Pi [1] et de l'application Android [2] avec le tag `MOC_SpeedTest`. Plusieurs tests ont été réalisés dans les deux sens, montant (Mobile vers Onboard) et descendant (Onboard vers Mobile).

Dans un premier temps, le but a été de transmettre des trames en continu et de tester les limites du système en comptabilisant les trames reçues. La MOC implémentée ne permet pas d'envoyer plus de 100 bytes par trame transmise. La longueur des trames et le délai entre chaque émission était modifiable. Il s'agit des tests `Down` et `Up`.

Les diagrammes de séquences ci-dessous présentent le déroulement des différents tests. La transmission du caractère '@' suivi d'un numéro permet à un périphérique d'annoncer au second qu'il va démarrer un test d'un certain type. A l'inverse, le caractère '#' indique la fin d'un test.

Down test

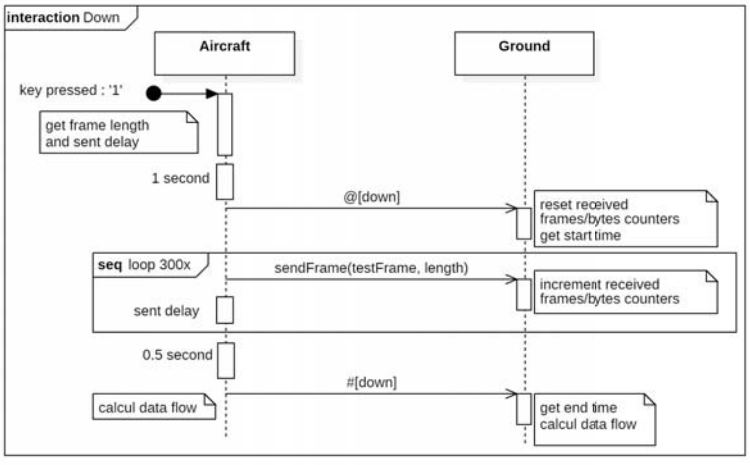


FIGURE 49: MOC Test - Down

1. Github Raspberry Pi : `https://github.com/jonathanmichel/Matrice210Pi`
2. Github Android : `https://github.com/jonathanmichel/Matrice210AndroidApp`

Le test `Down` est démarré depuis le système embarqué. Il va transmettre 300 trames d'une longueur définie à l'application Android à intervalle régulière. La longueur de la trame et l'intervalle sont choisies en début de test. Une fois la transmission terminée, le système embarqué indique à l'application que le test est fini et calcule son débit d'émission. L'application fait de même pour son débit de réception. Des délais sont ajoutés en début et fin de test afin de s'assurer que la précédente donnée a été correctement reçue et que le périphérique est dans un état stable.

De multiples tests ont été réalisés en variant les différents paramètres. Il a été constaté qu'il arrive que des trames soient perdues. Les résultats sont visibles dans le tableau de la figure 50. Le tableau complet se trouve en annexe. On constate que pour un taux de trame perdu raisonnable, le débit maximal avoisine le kilobytes/s.

Frame size [Bytes]	Delay [ms]	Lost frame [%]	Tx debit [B/s]	Rx debit [B/s]
100	13	40.00%	7692	4615
50	7	43.00%	7143	4071
25	4	42.00%	6250	3625
100	20	31.00%	5000	3450
50	15	19.00%	3333.5	2711
100	35	16.00%	2857	2410
100	50	3.00%	2000	1933
50	25	4.00%	2000	1913
100	70	2.00%	1429	1405
100	80	1.00%	1250	1238
100	100	1.00%	1000	993
50	50	1.00%	1000	993
10	10	2.00%	1000	977
10	10	8.00%	1000	920
100	150	0.00%	667	664
100	200	0.00%	500	498
50	500	0.00%	100	100
1	10	3.00%	100	97
1	25	2.00%	40	39

FIGURE 50: MOC Test - Down vitesses

La figure 51 montre l'interface de l'application Android à la fin d'un test `Down`. Dans ce cas, 300 trames de 100 bytes ont été transmises toutes les 80 millisecondes. On constate que seulement 297 trames ont été reçues, ce qui représente un taux de trame perdu de 1% pour un débit de réception de 1.2kBytes/s.

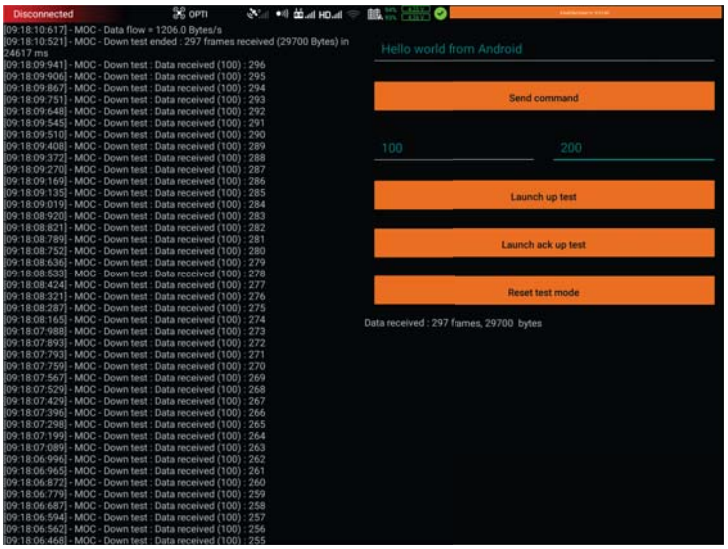


FIGURE 51: MOC Test - Down résultats

Up test

Un test similaire a été réalisé dans le sens montant mais les résultats n'étaient pas concluants, probablement dû à une erreur dans le code de l'application Android. Le MSDK retourne une erreur après une vingtaine de trames transmises, quel que soit la longueur de ces dernières.

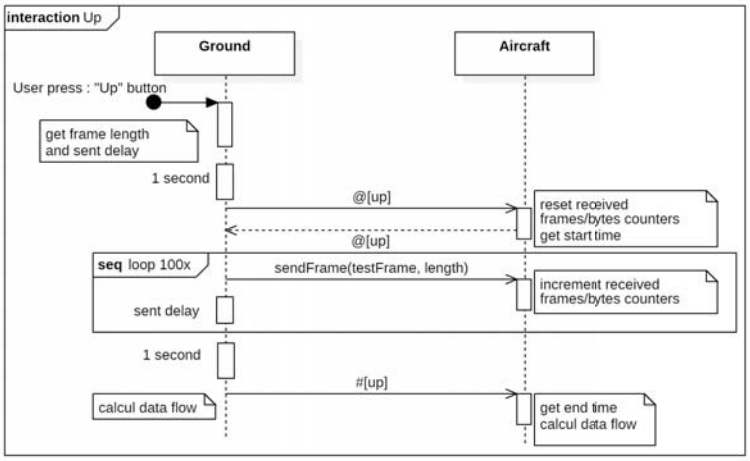


FIGURE 52: MOC Test - Up

Afin de tout de même pouvoir tester la communication montante, des tests avec acknowledge ont été mis au point. Il s'agit des tests `Ack-Down` et `Ack-Up`. Dans ces modes, le premier périphérique envoie un index au second. Ce dernier l'incrémente et renvoie une trame longue de 100 bytes dont les deux premiers bytes sont $-[index]$. Des vérifications sont faites des deux côtés pour vérifier que l'incrémentation se déroule correctement. Lorsque l'index atteint la valeur 100 le débit est calculé.

Ack-Down test

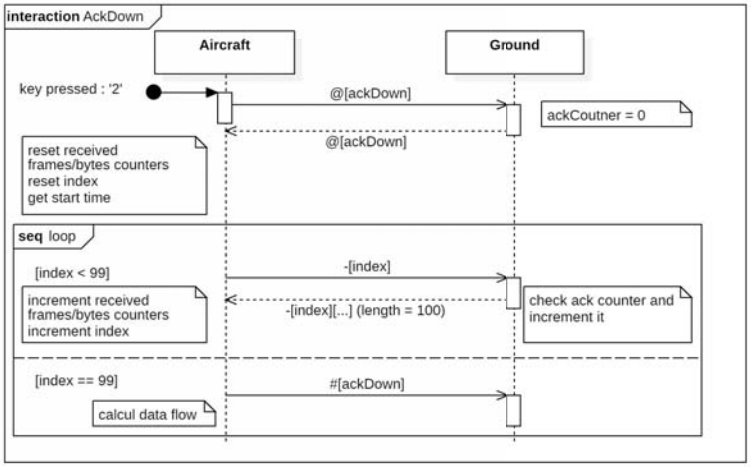


FIGURE 53: MOC Test - Ack-Down

Ack-Up test

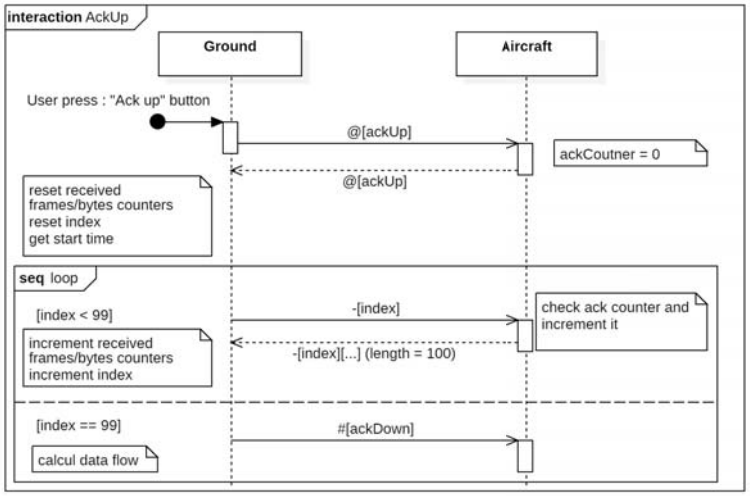


FIGURE 54: MOC Test - Ack-Up

Étant donné que le principe des tests ack est circulaire, il arrive qu'ils soient interrompus lorsqu'une trame est perdue. Les débits mesurés dans ces modes sont de l'ordre de 500Bytes/s en descendant (`Ack-Up`) et 400Bytes/s en montant (`Ack-Down`).

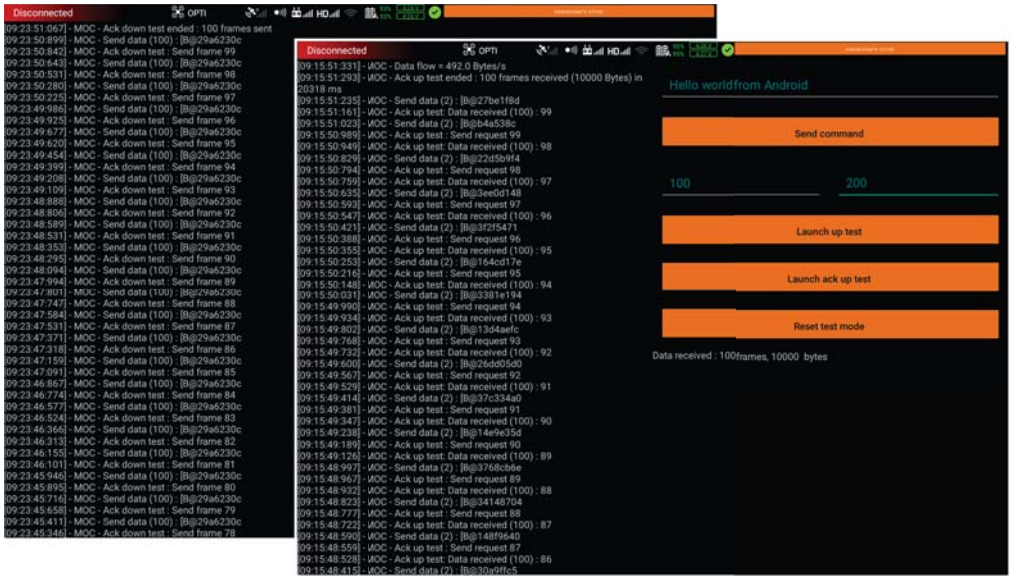


FIGURE 55: MOC Test - Ack-Up et Ack-Down résultats

On trouve ainsi majoritairement des débits inférieurs à ceux présentés par DJI qui, pour rappel, vente un transfert en up (Mobile vers Onboard) d'approximativement 1kB/s pour 8kB/s en down (Onboard vers Mobile). Il est difficile de déterminer d'où viennent ces différences tant il y a de couches entre le système embarqué et l'application Android, la plupart d'entre elles étant de plus inaccessibles pour l'utilisateur.

Étant donné que les informations sont transmises par UART entre le drone et l'ordinateur de bord, il se peut que ce soit cette interface qui pose problème. Les trames DJI pour la transmission de données concernant la MOC demandent 18 bytes pour les informations de header et de CRC[11]. Ainsi pour un baud rate de 115'200 Bauds en UART 8N1 et 100 bytes transmis via la MOC, le débit maximal en UART est de pratiquement 10kBytes/s, ce qui est largement supérieur au débit en jeu. La communication UART ne devrait donc pas poser problème.

$$UART_{8N1} \rightarrow 10bits/symb$$

$$UART = 115200bits/s = 11520symb/s$$

$$FrameDJI : header + CRC \rightarrow 18bytes$$

$$UART = 11520 \cdot 118 = 97frames/s$$

$$MOC = 97 \cdot 100 = 9700Bytes/s$$

Après discussion avec M. Pierre-André Mudry, il a été convenu qu'il était intéressant d'avoir essayé de pousser le système à ses limites mais pas nécessaire de mener des tests plus en détail. Les débits mesurés sont inférieurs à ceux attendus mais les résultats sont concluants pour que la communication Mobile-Onboard puisse être utilisée.

Il est à noter que lors de l'utilisation de l'IP Bridge, cf. chapitre 15, le taux de messages perdus augmente énormément. Il est difficile de dire précisément d'où vient le problème. Des études plus poussées auraient pu être faites en utilisant par exemple *Wireshark* pour analyser les trames IP transmises mais cela n'a pas de réel intérêt dans le cadre de ce travail de bachelor. Ce mode a simplement été évité lors des tests de vitesse.

15.2 Transfert

Il est important de préciser que les trames reçues via la MOC n'ont pas à être vérifiées par l'utilisateur, le *Onboard SDK Open Protocol* implémenté par DJI se charge de faire un contrôle d'erreur. Il se peut donc que des trames soient perdues, mais aucune trame fausse ne peut être reçue.

Le protocole permet également de préciser si le message transmis doit être suivi d'un ack ou non. Il s'agit d'une fonctionnalité de bas niveau non utilisée par DJI dans le cas d'un transfert de données en direction du MSDK. Même si cette dernière était utilisée, elle confirmerait uniquement la réception du message par le drone mais pas son transfert vers le MSDK.

S'il est nécessaire pour le développeur de garantir qu'une donnée a bien été transmise au MSDK en confirmant sa réception avec un ack, il se devra de créer son propre protocole avec acknowledge. Cette tâche n'a pas été réalisée dans le cadre de ce travail de bachelor car les performances de la MOC se sont révélées satisfaisantes à l'utilisation. Aucun problème n'a été constaté lors de l'envoi d'ordres au système embarqué pendant les différents tests. De même, le watchdog implémenté sur l'application Android envoie des ordres au système embarqué toutes les 500ms et fonctionne correctement.

16 Test du capteur embarqué

L'acquisition et la transmission des valeurs de l'antenne DVA fonctionnent correctement. Lorsqu'un émetteur DVA est à proximité de l'antenne la valeur transmise avoisine les 2000, lorsqu'aucun signal n'est présent elle vaut environ 100. Il s'agit de la valeur de l'ADC. Le traitement de cette valeur n'a pas été développé plus en détail, il le sera à l'avenir dans le cadre du produit que Vincent Bontempelli et moi-même souhaitons réaliser. Il en est de même pour le réglage de la sensibilité du boîtier récepteur de l'antenne. Cette dernière sera gérée par un composant externe indépendant de l'ordinateur de bord.

L'utilisation de l'antenne DVA dans le cadre de ce travail de bachelor permettait principalement de démontrer la possibilité de réaliser des mesures à distance avec le drone.

17 Tests GPS

Afin de tester les manipulations présentées au chapitre 11.11 sur les coordonnées GPS et l'ajout de vecteur NED, des tests unitaires ont été réalisés sur Qt. Le code est disponible sur Github [1]et chaque test y est détaillé. Le but est de tester les classes similaires à celles utilisées dans le logiciel déployé sur le Raspberry Pi et de se servir de *Google Earth* pour débugger ces dernières.

Comme expliqué au chapitre 11.11.2, il est possible pour l'utilisateur de faire tourner le plan XY du référentiel NED utilisé par le drone pour se déplacer. Ainsi l'axe X ne pointe plus forcément vers le Nord mais par exemple vers le Nord-Est si l'angle de rotation est de $+45°$. Le test implémenté pour vérifier le fonctionnement de cette partie fait tourner de $+60°$ le plan XY à six reprises et ajoute à chaque itération un vecteur $(100, 0)$ à un point C fixe. Le résultat est visible sur la figure 56, il s'agit d'un cercle de centre C et d'un rayon de 100 mètres. Tous les paramètres peuvent naturellement être changés ce qui a permis de faire plusieurs tests qui se sont tous révélés concluants.

FIGURE 56: Cercle généré par rotation de référentiel

La figure 57 présente un autre test où deux points $P1$ et $P2$ délimitent les deux coins opposés d'une zone carrée de recherche de victimes d'avalanche. Les coins manquants $P3$ et $P4$ sont ensuite calculés et l'axe X est défini le long des points $P1$ et $P4$. Les points X et Y sont ainsi ajouté avec respectivement des vecteurs $(100, 0)$ et $(0, 100)$ additionnés au point $P1$.

FIGURE 57: Avalanche sur le stade de Tourbillon

1. Tests unitaires GPS : https://github.com/jonathanmichel/Matrice210Qt

D'autres tests unitaires sont implémentés dans le code Qt et servent de vérification pour certains calculs utilisés dans les exemples ci-dessus.

18 Tests sur simulateur

Des tests réguliers ont été réalisés sur simulateur. Ils ont pu confirmer le bon fonctionnement des différentes parties software implémentées. Ainsi toutes les fonctionnalités développées sont disponibles à l'utilisation. Une attention particulière a été portée au comportement de l'arrêt d'urgence et du watchdog, aucune anomalie n'a été constatée pour ces éléments.

Des exemples basiques de tests sur simulateur sont présentés ci-dessous. Ils ont été choisis afin d'être synthétisables dans le cadre de ce rapport. En pratique, ils ont été répétés à maintes reprises en prenant soin de varier les valeurs utilisées.

Velocity Mission
La figure 58 montre un exemple de `Velocity Mission` où l'utilisateur demande au drone de se déplacer avec un vecteur de velocités de $(3, -4, 2)$. On constate dans le coin inférieur gauche du simulateur que ces valeurs sont correctement atteintes.

FIGURE 58: Velocity mission

Position Offset Mission

La figure 59 montre un exemple de deux `Position Offset Mission`. La première demande au drone d'avancer de 3 mètres (+3m en X) puis la seconde d'aller à gauche de 4 mètres (-4m en Y).

FIGURE 59: Position Offset mission

Waypoints Mission

La figure 60 montre un exemple de `Waypoints Mission`. Un premier passage a été fait manuellement avec la télécommande. A chaque changement de direction, la position du drone a été ajoutée à la liste de points de la mission. Le chemin a ensuite été rejoué de manière autonome par le drone.

FIGURE 60: Waypoints mission

19 Tests en extérieur

Des tests en extérieur ont été réalisés à trois reprises dans la cour de la Haute École d'Ingénierie, le but était de vérifier que le comportement obtenu sur simulateur ne diffère pas dans la réalité. Les tests ont toujours eu lieu à basse altitude et sur de courtes distances mais ils ont pu confirmer le bon déroulement du programme. Aucun problème particulier n'a été constaté.

Une vidéo d'un des tests présentant le déroulement d'une Waypoints Mission est disponible dans le répertoire de rendu sur le serveur. Une démonstration dans la cour de la HEI a été faite à M. Pierre-André Mudry juste avant le rendu de ce rapport.

FIGURE 61: Tests en extérieur

20 Améliorations futures

Dans l'état actuel du projet, certains points mériteraient d'être repensés et optimisés.

Il s'agit notamment de la gestion des différentes missions par le système embarqué. Il n'est actuellement possible d'exécuter qu'un seul type de mission à la fois. Il serait intéressant de pouvoir indiquer à l'ordinateur de bord que l'on souhaite donner une vélocité au drone puis le faire se déplacer d'un offset prédéfini avant de finalement lui demander de retourner à une position précise. Cette fonctionnalité sera nécessaire dans le cadre des missions pour le sauvetage des victimes d'avalanche actuellement en développement. Afin de mettre en place cette solution, l'architecture développée devra subir quelques modifications.

Les tests unitaires implémentés vérifient uniquement le bon fonctionnement de parties spécifiques du code du système embarqué et non de tout le système. Il serait nécessaire de tester le bon fonctionnement de la MOC en début d'exécution par exemple.

Actuellement lorsque le système embarqué demande le contrôle au drone et que ce dernier ne peut pas le lui accorder, la demande est répétée en continu et bloque l'utilisateur qui ne peut ni l'arrêter ni effectuer d'autres actions. Cette opération doit être repensée.

Les missions de décollage et d'atterrissage sont bloquantes et empêchent le traitement du watchdog, ces dernières devraient être déplacées dans le thread de la machine d'état du contrôleur de vol pour une meilleure efficacité.

L'interface de l'application Android est sommaire et peu ergonomique. Elle convient parfaitement dans le cadre de prototypage mais devra être améliorée pour une utilisation concrète. De même la gestion des accès au périphérique USB devrait être modifiée afin de garantir que la télécommande est bien connectée à l'application lorsqu'elle est démarrée. Il arrive qu'il soit nécessaire de débrancher et rebrancher le moniteur tactile pour indiquer à Android l'application à utiliser.

D'un point de vue hardware, la solution actuellement développée est un prototype. L'utilisation de l'antenne DVA comme capteur a demandé de nombreuses cartes à fixer sur le drone. Étant donné que cette solution est temporaire, peu de moyens ont été engagés pour cette tâche. A l'avenir une meilleure mécanique devra être conçue.

Plusieurs points devront être développés dans le cadre du projet d'entreprenariat de Vincent Bontempelli et moi-même. Afin de l'assister dans le développement de l'algorithme de recherche des victimes d'avalanches une carte des intensités mesurées sera par exemple implémentée sur le périphérique mobile.

Cinquième partie
Conclusion

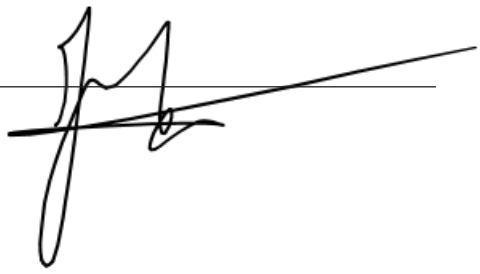
Les buts du projet sont atteints. Le système embarqué a correctement été implémenté sur le drone et l'application Android au sol permet à l'utilisateur d'interagir avec l'ordinateur de bord. La transmission d'informations du capteur embarqué jusqu'au sol en temps réel est fonctionnelle. Les débits mesurés sont inférieurs à ceux promis par le constructeur mais les performances suffisent amplement pour que la communication de données soit utilisable. Des fonctionnalités de vols autonomes ont été implémentées afin de proposer un démonstrateur des possibilités offertes par le Onboard SDK utilisé sur le drone.

Étant donné que ce projet s'inscrit dans une démarche entrepreneuriale, il est très encourageant de constater que le système est fonctionnel. Cela permet de se rendre compte que l'utilisation de ce drone est envisageable pour le sauvetage. De nombreux points sont encore à développer avant de pouvoir localiser des victimes d'avalanches mais la technologie et les avancements faits durant ce travail de bachelor sont concluants.

21 Remerciements

Pour terminer, je tiens à remercier M. Pierre-André Mudry d'avoir accepté de me suivre sur ce projet mais également de m'avoir soutenu et conseillé. L'institut Systèmes industriels pour l'acquisition du drone lorsque Vincent Bontempelli et moi avons fait part de notre projet. M. Marc Pignat pour son aide sur Linux et M. Darko Petrovic pour son partage d'expérience avec Zephyr Os.

Jonathan Michel
Sion, le 24 août 2018



Sixième partie

Annexes

A Glossaire

- **API** Application Programming Interface
- **DVA** Détecteur de Victime d'Avalanche
- **HEI** Haute École d'Ingénierie de Sion
- **I/O** Input/Output
- **I2C** Inter-Integrated Circuit
- **MOC** Mobile-Onboard Communication
- **MTOM** Maximum Take-Off Mass
- **MSDK** Mobile Software Development Kit
- **NED** North-East-Down Coordinates
- **OB** Onboard
- **OS** Operating System
- **OSDK** Onboard Software Development Kit
- **ROS** Robot Operating System
- **RTH** Return-to-home
- **RTK** Real Time Kinematic
- **RTOS** Real-Time Operating System
- **SDK** Software Development Kit
- **SPI** Serial Peripheral Interface
- **UI** User Interface
- **UX** User eXperience
- **UXSDK** User eXperience Software Development Kit

B Liste de figures

Table des figures

C Sources

Références

1. Office fédéral de l'aviation civile OFAC. Drones et modèles réduits (Consulté le 13 août 2018). 2018. https://www.bazl.admin.ch/bazl/fr/home/bonasavoir/drones-et-modeles-reduits.html.
2. Office fédéral de l'aviation civile OFAC. Carte pour drones (Consulté le 13 août 2018). 2018. https://www.bazl.admin.ch/bazl/fr/home/bonasavoir/drones-et-modeles-reduits/drohnenkarte.html.
3. DJI. Fly Safe Geo Zone Map (Consulté le 12 juin 2018). 2018. https://www.dji.com/flysafe/geo-map.

4. Office fédéral de l'aviation civile OFAC. Règles et informations sur les drones (Consulté le 13 août 2018). 2018. https://www.bazl.admin.ch/bazl/fr/home/bonasavoir/drones-et-modeles-reduits/regles-informations-drones.html.
5. skyguide. Vols spéciaux (Consulté le 13 août 2018). 2018. https://www.skyguide.ch/fr/services/vols-speciaux/.
6. Office fédéral de l'aviation civile OFAC. Autorisations d'exploiter des drones à proximité d'un rassemblement de personne (Consulté le 13 août 2018). 2018. https://www.bazl.admin.ch/bazl/fr/home/bonasavoir/drones-et-modeles-reduits/autorisations-d_exploiter-des-drones-au-dessus-dun-rassemblement.html.
7. skyguide. Swiss U-Space (Consulté le 13 août 2018). 2018. https://www.skyguide.ch/fr/evenements-medias/u-space-live-demonstration/.
8. DJI. Matrice 200 Serie Specs (Consulté le 12 juin 2018). 2018. https://www.dji.com/matrice-200-series/info#specs.
9. DJI. Mobile - Onboard SDK communication (Consulté le 12 juin 2018). 2018. https://developer.dji.com/onboard-sdk/documentation/guides/component-guide-mobile-communication.html.
10. Zephyr Project. C++ Support for Applications (Consulté le 12 juin 2018). 2018. http://docs.zephyrproject.org/kernel/other/cxx_support.html.
11. DJI. Onboard SDK Open Protocol (Consulté le 21 août 2018). 2018. https://developer.dji.com/onboard-sdk/documentation/introduction/index.html.

D Documents

1. Demande pour la coordination des activités aériennes spéciales
2. Pré-autorisation pour activités aériennes spéciales
3. Choix du RTOS et de la carte embarquée
4. Schéma de connexion complet
5. Ticket SInf Wifi device-hevs
6. Mesures communication Mobile-Onboard
7. Datasheet Antenne DVA DAS000

Les documents annexés ci-après sont titrés et numérotés. Les originaux sont disponibles dans le répertoire de rendu sur le serveur.

Demande pour la coordination des activités aériennes spéciales

1. Informations concernant le demandeur

Date de la demande

Téléphone

Nom/Société

E-mail

Adresse

Numéro de contact
(durant l'activité)

2. Informations concernant l'activité

Type d'activité

Heure(s), (locale)

Lieu exact

Date(s)

Durée

3. Informations concernant les vols spéciaux avec aéronefs habités

Immatriculation /
Indicatif d'appel

Règles de vol

IFR

VFR

Type d'aéronef

Altitude

Document 1 - Demande pour la coordination des activités aériennes spéciales

4. Informations concernant les lanternes célestes (avec un volume de 1m3 ou moins), les ballons pour enfants, les feux d'artifice, lasers et faisceaux lumineux

(Pour les lanternes célestes et les ballons pour enfants avec un volume supérieur à 1m3, le paragraphe 5 doit être complété)

Nombre

Lâcher

Charge (objets attachés)

Grappe de ballons?

Oui

Non

Direction du faisceau lumineux ou du laser

Altitude du feu d'artifice (en mètres)

5. Informations concernant les activités aériennes sans occupants

Type d'aéronef sans occupants

Base légale

La déclaration ci-dessous s'applique uniquement aux demandeurs d'activités citées dans l'Ordonnance du DETEC sur les aéronefs de catégories spéciales (OACS, SR 748.941) articles 15, 16 et 17.
Ces activités sont les suivantes:
modèles réduits d'aéronefs (mini-drones inclus);
cerfs-volants, parachutes ascensionnels et ballons captifs;
ballons libres et lanternes lumineuses avec une capacité supérieure à 1m3.

Coordonnées du centre de la zone d'activité

Rayon autour du centre (en mètres)

Altitude (en mètres sol)

Déclaration concernant la sécurité des tiers au sol

En adressant cette demande, le demandeur des opérations des aéronefs sans occupants ou autres activités aériennes déclare que:
1. elle/il a pris connaissance des règles de l'Ordonnance du DETEC sur les aéronefs de catégories spéciales (OACS, SR 748.941), notamment l'article 20 concernant l'éventuelle exigence de souscription d'une assurance responsabilité civile et l'article 19 concernant les éventuelles prescriptions cantonales.
2. ces activités aériennes ne sont pas conflictuelles avec les lois et régulations nationales en vigueur concernant la protection des données privées et la protection des installations militaires.
3. elle/il a pris les actions nécessaires afin qu'aucun tiers au sol ne soit mis en danger par les activités aériennes sous son contrôle.

Lieu / Date

Prénom / Nom

Cette demande ne sera pas traitée si les cases "lieu et date", ainsi que "prénom et nom" ne sont pas remplis..

Document 1 - Demande pour la coordination des activités aériennes spéciales

Mémorisez le formulaire complété et envoyez-le à l'adresse suivante:

specialflight@skyguide.ch

Informations importantes

Les demandes complétées doivent parvenir au guichet d'informations au moins dans les 10 jours ouvrables avant le premier jour d'opération. Les demandes tardives ne peuvent pas être prises en considération. Les autorisations finales pour les activités aériennes ou vols spéciaux ne peuvent être octroyées que le jour du vol par l'unité de contrôle de la circulation aérienne responsable. Les permissions délivrées par skyguide concernent uniquement l'accès à l'espace aérien. Skyguide se réserve le droit de refuser cette demande ou d'imposer des conditions ou restrictions supplémentaires si nécessaire.

De plus amples informations concernant les activités aériennes ou vols spéciaux nécessitant une coordination sont disponibles sur le site internet skyguide.

Document 2 - Pré-autorisation pour activités aériennes spéciales

to Haute École d'Ingénierie de Sion, Jonathan Michel
e-mail jonathan.michel@students.hevs.ch
phone

from Special Flight Office, AMC
Flugsicherungsstrasse 1-5
CH-8602 Wangen bei Dübendorf
specialflight@skyguide.ch
043 931 62 36

date Wangen bei Dübendorf, 17.07.2018

Pré-autorisation pour activités aériennes spéciales

Ce document ne constitue en aucun cas une autorisation définitive pour le déroulement de l'activité. L'autorisation finale ne pourra être délivrée que le jour de l'activité par l'organe du contrôle de la circulation aérienne concerné.

A. Activité coordonnée			
Numéro de référence:	**SUA18-1561**		
Periode de validité de:	2018-07-30	**à:**	2018-09-07
Horaire:	0800 - 1800 LT	**Type d'activité:**	Test flight
Lieu / Région:	Sion	**Coordonnées:**	46°14′24.635″N 7°21′31.085″E
Radius:	40m	**Altitude / Niveau:**	max. 40m AGL
Immatriculation / Indicatif d'appel:		**Type d'aéronef:**	RPAS

B. Procédure de notification

- Le jour de l'activité, 1 heure avant le début du vol ou de l'activité, prendre contact avec le superviseur Sion Tower par téléphone au +41 58 461 21 33, en indiquant le numéro de référence.

C. Conditions et restrictions

- Si une activité annoncée devait être annulée, le superviseur Sion Tower doit en être informé immédiatement.
- A la fin de l'activité, le responsable doit contacter le superviseur de la tour de contrôle pour l'informer de la fin de l'activité.
- Votre activité de mini-drone doit se dérouler à l'intérieur de la zone d'activité sollicitée.
- La/Le pilote du mini-drone doit appliquer le principe "voir et éviter", selon les règles suivantes :
  - 〉 être en mesure de prendre les actions nécessaires en tout temps et en toute circonstance afin d'éviter des collisions avec des aéronefs non-participants
  - 〉 constamment avoir un contact visuel direct avec le mini-drone
- Les activités des mini-drones doivent céder la priorité aux autres utilisateurs de l'espace aérien (par exemple avion, hélicoptère, etc).

Document 2 - Pré-autorisation pour activités aériennes spéciales

- Vous êtes prié de faire atterrir le drone immédiatement si un avion s'approche.
- Si vous perdez le contrôle de votre mini-drone, nous vous prions d'en informer immédiatement le superviseur de la tour de contrôle.
- La personne de contact doit être joignable par téléphone durant toute la durée de l'activité. Les informations détaillées du contact (nom, n° de téléphone) doivent être confirmées avec le Superviseur Sion Tower lors de la coordination avant le début de l'activité.
- Ce préavis n'est valable que pour l'utilisation de l'espace aérien. Toute(s) autorisation(s) éventuelle(s) requise(s) par d'autres instances suisses (OFAC, police cantonale, commune, etc…) demeure(nt) de la responsabilité du demandeur. Si l'activité est prévue de se dérouler au-delà du territoire Suisse, il est de la responsabilité du demandeur de s'assurer d'être en possession de toutes les autorisations requises par le pays concerné.
- Informez skyguide par e-mail à specialflight@skyguide.ch, si l'activité coordonnée :
  - ⟩ doit être modifiée
  - ⟩ doit être prolongée
  - ⟩ doit être annulée
- En tout temps, skyguide se réserve le droit de refuser, d'interrompre, de restreindre ou d'annuler l'activité pour des raisons opérationnelles.

Document 3 - Choix du RTOS et de la carte embraquée

1 RTOS

Le choix d'un RTOS est crucial. Le but n'est pas de devoir passer du temps à assurer un portage sur une carte spécifique. Il est primordial de choisir un OS avec une communauté active, un projet en cours de développement et qui a de l'avenir.

FreeRTOS a été présenté en cours et est très répandu dans l'industrie. Il répond aux critères ci-dessus. Néanmoins il est assez particulier, pas très agréable à utiliser et essuie de nombreuses critiques au sein des professeurs de la HEI. Il a d'ailleurs été abandonné cette année au profit de *Keil RTX* pour les prochains cours de PTR (Programmation Temps Réel).

Keil RTX est proposé par l'entreprise du même nom et nécessite l'utilisation de l'IDE propriétaire. Celui-ci est payant, uniquement disponible sur Windows et propose une ergonomie limitée. La HEI dispose de licences mais dans une optique de pérennité, de portabilité et afin de favoriser l'open-source, cette solution a été écartée.

Le choix s'est tourné vers *Zephyr OS*. Il s'agit d'un système d'exploitation open-source léger, conçu pour les appareils aux ressources limitées et supportant plusieurs architectures. Il est actuellement en développement et fait partie de la *Linux Foundation*. Le projet est entre autre supporté par *Intel*, *NXP Semiconductor*, *Nordic Semiconductor* et *Linaro*. L'objectif du projet *Zephyr* est de devenir le meilleur RTOS open source pour les appareils à ressources limitées.

Il existe des dixaines de RTOS et il a fallu faire un choix. D'autres OS ont été découvert mais ne sont pas présentés ici. *Zephyr* répondant parfaitement aux critères et n'ayant pas de désavantages il a été sélectionné.

Il est important de noter que tous les RTOS cités ci-dessus ont un kernel en C. C'est le langage de prédilection pour cette application. Les systèmes d'exploitation proposés en C++ sont rares et peu utilisés. On peut citer en exemple *distortos* qui est en cours de développement par une demi-douzaine de développeurs selon leur *Github*. Ne répondant pas aux critères présentés en début de chapitre, ce projet et ses semblables ont été écartés.

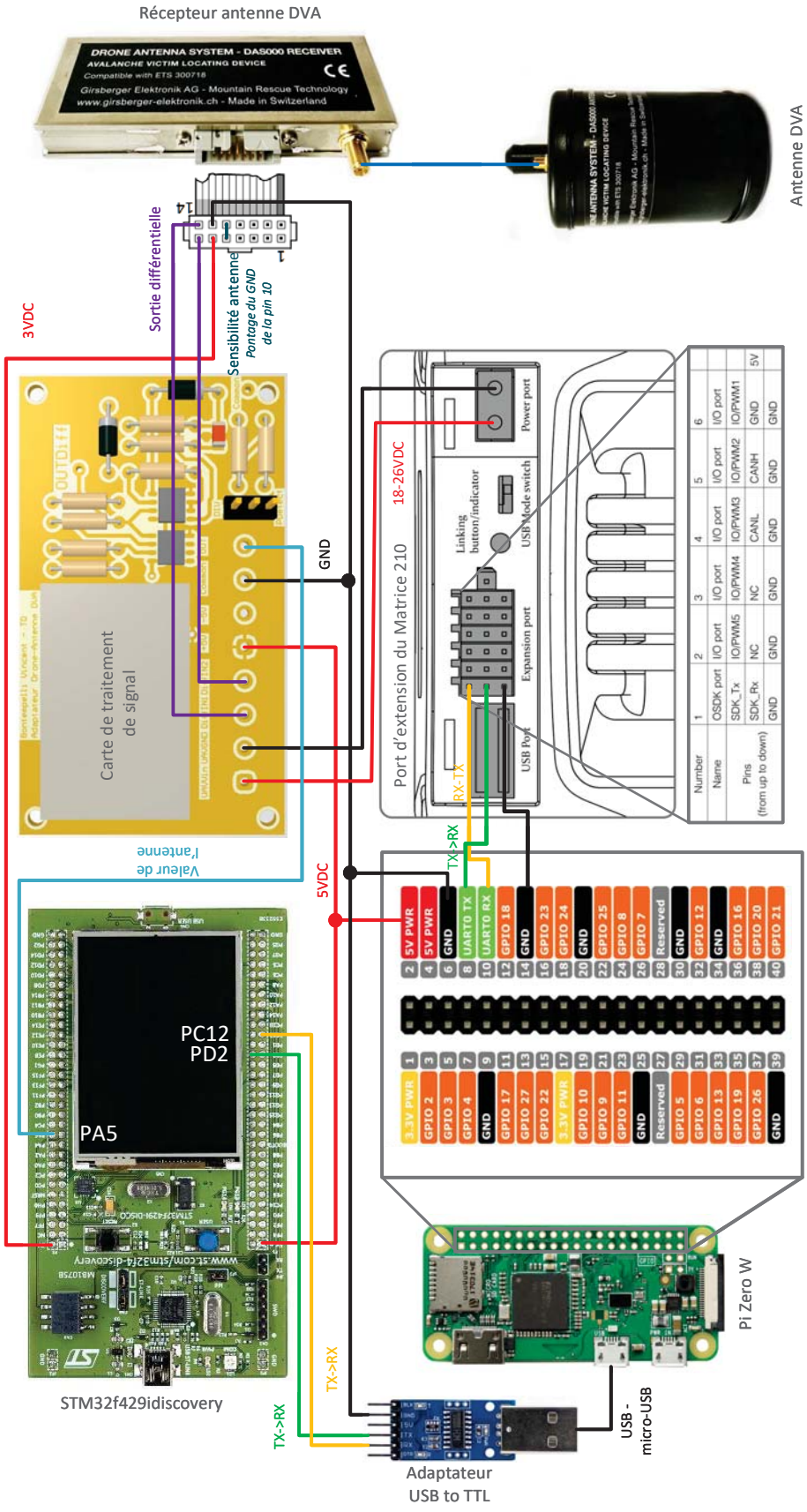
Le SDK fournit par DJI est lui codé en C++. Il n'est pas impossible de combiner les deux langages mais cela demande quelques adaptations.

2 Carte de développement

Une fois l'OS déterminé, il a été choisi de prendre une carte de développement *Nucleo*. Ces dernières sont très utilisées et leur support est bien avancé pour *Zephyr Os*. En particulier pour la carte *Nucleo-L476RG* qui a été commandée dans un premier lieu, les fonctionnalités d'I2C et de SPI étant notamment implémentées ce qui pourra servir pour le futur système de mesure externe. Il s'agit d'une carte basée sur un *STM32L476RG*, un Cortex-M4 32-bit tournant jusqu'à 80 MHz. Elle dispose de 1 Mbyte de flash et 128 Kbyte de SRAM.

A noter que le choix de la carte n'est pas primordial, *Zephyr* met à disposition de nombreux niveaux d'abstraction ce qui permet de définir la cible à la compilation et de minimiser les adaptations nécessaires suite à un changement de cette dernière. Pour ce faire, le projet fonctionne avec Device tree et des fichiers de configuration permettant de décrire le hardware utilisé.

Document 4 - Schéma de connexion complet



Document 5 - Ticket SInf Wifi device-hevs

De : Service informatique
Envoyé le :vendredi, 6 juillet 2018 15:56
À : Jonathan Michel
Objet :[GLPI #0016504] Nouveau suivi Connecter au réseau Wi-Fi ou RJ un Raspberry Pi Zero

=-=-=-= Pour répondre par courriel, écrivez au dessus de cette ligne =-=-=-=

Bonjour, vous avez reçu un nouveau suivi pour le ticket No: 0016504 concernant

Connecter au réseau Wi-Fi ou RJ un Raspberry Pi Zero

Votre Raspberry Pi Zero dont vous nous avez donné l'adresse MAC peut maintenant se connecter. Voici les paramètres.

Réseau Wi-Fi (SSID) :
device-hevs
Type de cryptage :
WPA2-PSK (AES)
Clef de cryptage (au format texte):
[redacted]

Ca devrait correspondre à ça dans le fichier
/etc/wpa_supplicant/wpa_supplicant.conf :

network={
ssid="device-hevs"
psk="[redacted]
proto=RSN
key_mgmt=WPA-PSK
pairwise=CCMP
auth_alg=OPEN
}

MERCI DE ME DONNER UN FEED-BACK si c'est OK (ou non !) en répondant à ce message.

Salutations,
Pour le SInf.

Si vous deviez répondre directement à cet e-mail, nous vous prions de cliquer sur répondre et d'effacer le texte des mails précédents. Cela facilitera le traitement automatique de votre e-mail par notre système.

--
HES-SO Valais Wallis - Service Informatique
Généré automatiquement par GLPI 0.85.5

=_=_=_= Pour répondre par courriel, écrivez au dessous de cette ligne =_=_=_=

Be green, keep it on the screen

MOC - Down test

UART [Bd]	To send [frame]	Frame size [Bytes]	Delay [ms]	Received [frames]	Lost frame [%]	Tx debit [frames/s]	Tx debit [B/s]	Rx debit [B/s]
115200	300	100	50	290	3.00%	20	2000	1933
115200	300	1	10	289	4.00%	100	100	96
115200	300	10	10	281	6.00%	100	1000	937
115200	300	75	30	279	7.00%	33.33	2499.75	2325
115200	300	50	20	268	11.00%	50	2500	2233
115200	300	50	20	265	12.00%	50	2500	2208
115200	300	10	5	253	16.00%	200	2000	1687
115200	300	100	20	217	28.00%	50	5000	3617
230400	300	100	150	299	0.00%	6.67	667	664
230400	300	100	200	299	0.00%	5	500	498
230400	300	50	500	299	0.00%	2	100	100
230400	300	100	80	297	1.00%	12.5	1250	1238
230400	300	100	100	298	1.00%	10	1000	993
230400	300	50	50	298	1.00%	20	1000	993
230400	300	100	70	295	2.00%	14.29	1429	1405
230400	300	10	10	293	2.00%	100	1000	977
230400	300	1	25	295	2.00%	40	40	39
230400	300	100	50	290	3.00%	20	2000	1933
230400	300	1	10	290	3.00%	100	100	97
230400	300	50	25	287	4.00%	40	2000	1913
230400	300	10	10	276	8.00%	100	1000	920
230400	300	100	35	253	16.00%	28.57	2857	2410
230400	300	50	15	244	19.00%	66.67	3333.5	2711
230400	300	100	20	207	31.00%	50	5000	3450
230400	300	100	13	180	40.00%	76.92	7692	4615
230400	300	25	4	174	42.00%	250	6250	3625
230400	300	50	7	171	43.00%	142.86	7143	4071

MOC - Down test - Frame duration

UART [Bd]	Frame size [Bytes]	Delay [ms]	Lost frame [%]	Tx debit [B/s]	Rx debit [B/s]	UART Frame duration [ms]	Frame duration < Delay ?
115200	100	50	3.00%	2000	1933	10.24	Yes
115200	1	10	4.00%	100	96	1.65	Yes
115200	10	10	6.00%	1000	937	2.43	Yes
115200	75	30	7.00%	2499.75	2325	8.07	Yes
115200	50	20	11.00%	2500	2233	5.90	Yes
115200	50	20	12.00%	2500	2208	5.90	Yes
115200	10	5	16.00%	2000	1687	2.43	Yes
115200	100	20	28.00%	5000	3617	10.24	Yes
230400	100	150	0.00%	667	664	5.12	Yes
230400	100	200	0.00%	500	498	5.12	Yes
230400	50	500	0.00%	100	100	2.95	Yes
230400	100	80	1.00%	1250	1238	5.12	Yes
230400	100	100	1.00%	1000	993	5.12	Yes
230400	50	50	1.00%	1000	993	2.95	Yes
230400	100	70	2.00%	1429	1405	5.12	Yes
230400	10	10	2.00%	1000	977	1.22	Yes
230400	1	25	2.00%	40	39	0.82	Yes
230400	100	50	3.00%	2000	1933	5.12	Yes
230400	1	10	3.00%	100	97	0.82	Yes
230400	50	25	4.00%	2000	1913	2.95	Yes
230400	10	10	8.00%	1000	920	1.22	Yes
230400	100	35	16.00%	2857	2410	5.12	Yes
230400	50	15	19.00%	3333.5	2711	2.95	Yes
230400	100	20	31.00%	5000	3450	5.12	Yes
230400	100	13	40.00%	7692	4615	5.12	Yes
230400	25	4	42.00%	6250	3625	1.87	Yes
230400	50	7	43.00%	7143	4071	2.95	Yes

Frame DJI 18 bytes for header and CRC

GIRSBERGER
Mountain Rescue Technology

Data Sheet

DAS000 Drone Antenna System 457kHz

The DAS000 enables a very fast search by drone or UAV of persons who have become buried in an avalanche, given that the persons are equipped with a transceiver that operates on the standard frequency of 457 kHz.

The system consists of an antenna and a receiver. The antenna is attached under the drone with a distance which has to be determined. This avoids any exposure to electromagnetic interference from the drone. The receiver is mounted in the drone and connected by cable connection to the antenna and to the control of the drone.

The received avalanche beacon signal is converted in the receiver into an LF signal. The signal strength is increased as approaching to the victim or decreases with increasing distance. This NF signal is used to control the nine levels of the receiver sensitivity and to control the drone. The system exhibits the same range (approximately 40 m) in all directions, it cannot provide any information about the direction of the magnetic field lines.

For the operation of the receiver is an interface for power supply and for the control of the receiving sensitivity available.
A separate and clean power supply is an essential requirement.
The software for the control of the receiver and the drone is not part of this system.

Receiver

Connections and Level Adjustment

Antenna connection SMA
Interface to the drone, Ribbon cable connector
Level adjustment, LF signal

Girsberger Elektronik AG :: Mountain Rescue Technology :: Oberdorfstrasse 7 :: CH-8416 Flaach
Tel. 0041 (0) 52 301 35 35 :: www.girsberger-elektronik.ch

GIRSBERGER
Mountain Rescue Technology

Ribbon cable connector pin assignment

Pin	Name	Signal	Operating Mode / Comments
1	ATT 1	Control input connected to GND	Receive (search) mode, highest sensitivity
2	ATT 2	Control input connected to GND	
3	ATT 3	Control input connected to GND	
4	ATT 4	Control input connected to GND	
5	ATT 5	Control input connected to GND	
6	ATT 6	Control input connected to GND	
7	ATT 7	Control input connected to GND	
8	ATT 8	Control input connected to GND	
9	ATT 9	Control input connected to GND	Receive mode, lowest sensitivity
10	GND	Ground (0 V) for control inputs	
11	UB	Supply voltage + 3.0 VDC	observe correct polarity!
12	GND	Ground (0 V)	
13	OUT 1 -	Output LF 2 kHz	Galvanically separated with level adjustment
14	OUT 2 +	Output NF 2 kHz	Galvanically separated with level adjustment

Technical Data

RX frequency:	457 kHz +/- 100 Hz (international standard)
Power supply:	+ 3.0 VDC
Power consumption:	13 mA
Minimum/maximum sensitivity:	50 nV / 1 µV
LF Output, galvanically separated:	2 kHz
Operating temperature range:	- 20° C to + 40° C
Dimensions:	119 x 42 x 19 mm
Weight:	84 g
Protection class:	no protection
Compatibility:	ETS 300718

Antenne

Technical Data

Frequency range:	457 kHz +/- 100 Hz (international standard)
Polarization:	omnidirectional 360°
Power supply:	from receiver
Operating temperature range:	- 20° C to + 40° C
Dimensions:	Length 112 mm, diameter 75 mm
Weight:	240 g
Protection class:	IP 67

We reserve the right to change identifiers, dimensions and mechanical configurations at any time.

Ref. 20160518

Girsberger Elektronik AG :: Mountain Rescue Technology :: Oberdorfstrasse 7 :: CH-8416 Flaach
Tel. 0041 (0) 52 301 35 35 :: www.girsberger-elektronik.ch