Drone 1 D'imagerie et de Télémétrie

Académie Aix Marseille
Lycée Alphonse Benoit
Cours Victor Hugo
84800 L'Isle sur la Sorgue

IR 1 : Carrillo Thomas
IR 2 : Busseret Dylan
EC 1 : Lozano Dorian
EC 2 : Buffe Jordan

Sommaire :

Présentation Générale

Introduction :

Il est maintenant devenu courant d'utiliser un drone pour effectuer des prises de vue ou des séquences filmées des maisons ou de paysages.
Un besoin supplémentaire apparaît dans l'industrie, celui de pouvoir effectuer des mesures afin de surveiller, détecter, maintenir des ouvrages hauts et difficiles d'accès de manière préventive.

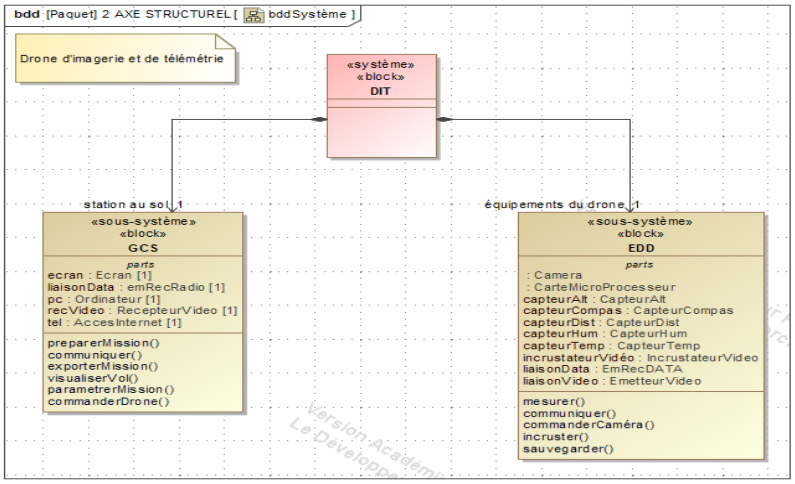
Les architectes-maitres d'œuvre ont besoin lors des phases de construction ou de maintenance d'observer les murs et toitures des bâtiments, vérifier les montages d'huisseries, de faitières, rechercher des entrées d'eau etc...

L'accès à ces zones n'est pas toujours possible pour les grands ouvrages (ponts, tours destinées au logement etc...). L'utilisation d'un drone d'observation équipé d'une transmission vidéo, d'un appareil photo ainsi que d'une batterie de capteurs des grandeurs physiques est alors une solution économique et efficace.

Le matériel de prise de vue et de vidéo habituellement utilisé par les professionnels est un CANON 5D dont le zoom et la prise de vue sont commandés électriquement par bus USB. L'adaptation d'un appareil photo léger et à faible coût de type Camera GOPRO permet de proposer aux clients une solution économique de la surveillance avec néanmoins une qualité vidéo suffisante.

Ainsi, en ajoutant des capteurs à un drone, il se révèle un atout majeur et performant pour la maintenance de ces ouvrages.
Ce système nécessite deux personnes. Le pilote depuis la station au sol (GCS) s'occupe du pilotage à vue du drone. Un technicien (copilote) à ses côtés supervise la prise d'images, la télémétrie et la communication avec l'entreprise.

Cahier Des Charges :



Le système comprend deux parties :

Les équipements du drone (EDD).
La station de contrôle au sol (GCS, Ground Control Station, acronyme anglais usuel).

Description des besoins de L'EDD :
La durée d'un vol peut varier entre 15 et 30 minutes.

En raison de la taille de certains ouvrages, une mesure de distance d'objet est indispensable. Le drone est équipé d'un capteur GPS, d'un altimètre, d'un capteur d'humidité, de température ambiante, de distance infrarouge.

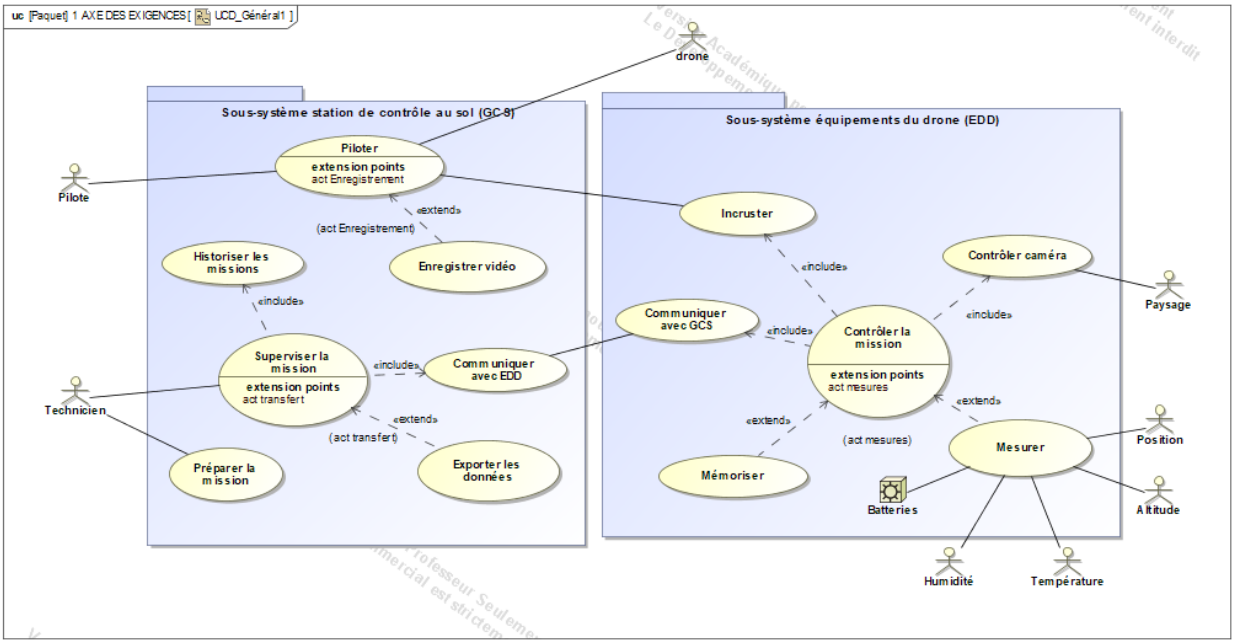
Le copilote (technicien) envoie au drone des ordres pour prendre des photos ou des séquences vidéo qui seront stockées dans la carte mémoire de l'appareil photo.
Sur un ordre de la GCS, le drone effectue des mesures, les transfère à la station au sol, les sauve également dans sa mémoire en cas de difficultés de communication.
Chaque photo prise sera associée aux mesures prises.
Le drone émet l'image de la caméra vers un écran de la station au sol avec la possibilité d'incruster des paramètres (texte, mesures, etc.) au choix du pilote.

Description des besoins du GCS :
La station au sol comprend un écran de contrôle associé à une radio commande pour piloter le drone.
Une base de données permettant de mémoriser les données des vols.
Un logiciel sur ordinateur permet :

- De préparer la campagne de vol.
- D'émettre des ordres de prises de photos ou de vidéos.
- D'effectuer l'acquisition des mesures en temps réel.
- De modifier les données à incruster dans le retour vidéo.
- De transmettre les données vers la base de données.

Spécification :

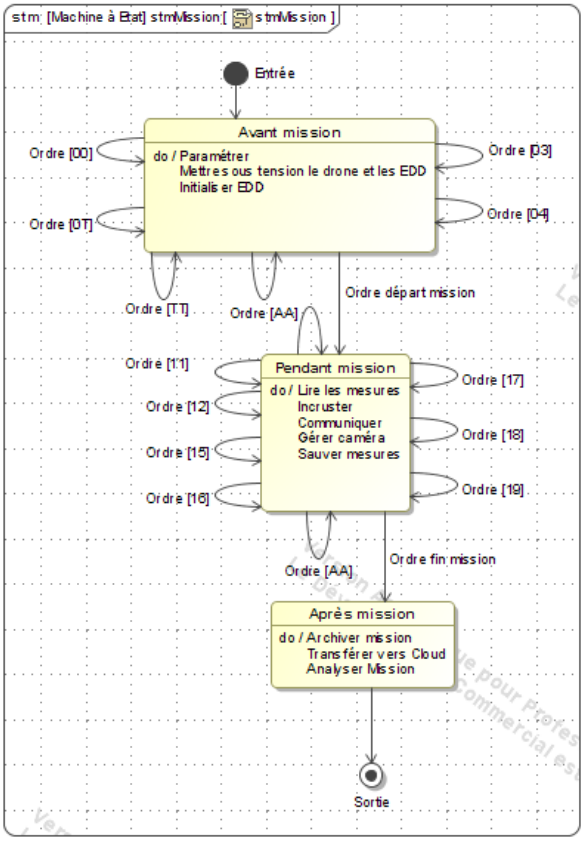
Les besoins du système sont représentés par le diagramme des cas d'utilisation suivant :



Durant son fonctionnement, le système peut se trouver dans un des trois états suivants :

- Préparation : état « avant mission ».
- Mission de vol : état « pendant mission ».
- Récupération et exportation des données de mission : état « après mission ».

Pour chaque état, un protocole de communication est établi.
Les fonctions (ordres) de ce protocole sont accessibles selon l'état du système.
Le diagramme d'états ci-contre permet d'identifier les ordres disponibles en fonction de l'état du système.



IR 1 - Communication avec GCS et EDD

Présentation générale de la partie communiquer :

Ma partie concerne la communication entre l'équipe au sol (GCS) et le drone (EDD). En effet, pour mener à bien le projet, l'EDD et la GCS doivent impérativement pouvoir communiquer. Les tâches suivantes m'ont donc été attribuées :

Étudiant	*Liste des fonctions assurées par l'étudiant*	*Installation* : Système Linux embarqué et chaîne de compilation croisée. *Mise en œuvre* : Communication liaison DATA, chiffrement/déchiffrement. *Configuration* : Modules de communication, environnement graphique. *Réalisation* : Classe C++ de communication et de gestion du protocole DATA, programme principal. *Documentation* : Plan d'intégration des différentes parties logicielles, élaboration partielle du document « en cas d'anomalie de fonctionnement »
IR1 IR	EDD+GCS : UCs Communiquer avec GCS et EDD	

Avec mon équipe nous avons donc décidé d'attribuer les tâches entre nous de manière prévisionnelle accompagnées du temps qu'elles nous prendraient. Pour ma part :

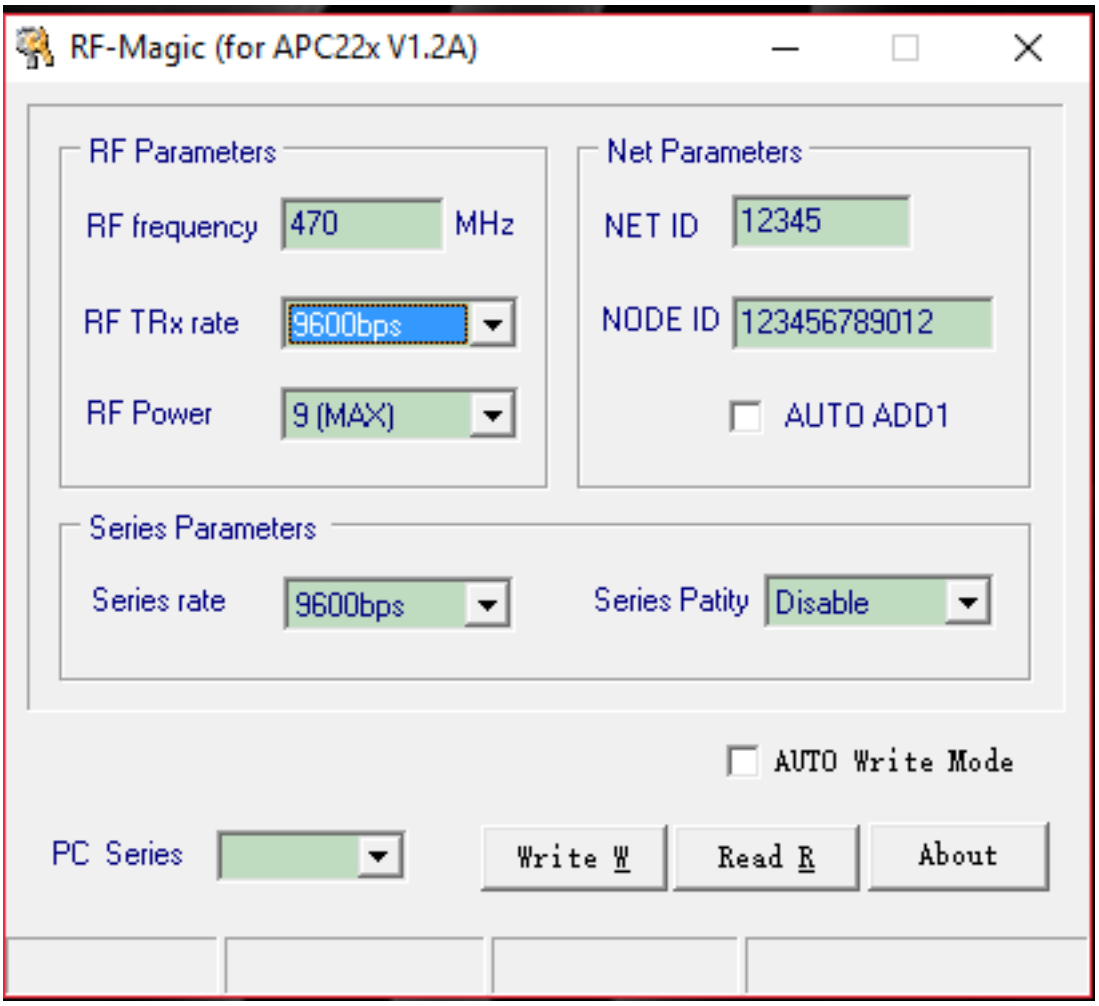
	Tâches Carrillo	204 heures	Jeu 11/01/18	Ven 25/05/18		CARRILLO Thomas
	Recherche	20 heures	Jeu 18/01/18	Ven 26/01/18	7	
	codage de la communication DATA (et envoi des mesures)	50 heures	Ven 26/01/18	Mer 14/03/18	26	
	Cryptage des échanges	10 heures	Mer 14/03/18	Ven 16/03/18	27	
	Mise en oeuvre de la communication par liaison radio	30 heures	Ven 16/03/18	Jeu 29/03/18	28	
	Analyse et codage du protocole DATA	60 heures	Jeu 29/03/18	Mer 16/05/18	29	
	Intégration du CRC et cryptage dans les échanges radio	10 heures	Mer 16/05/18	Ven 18/05/18	30	
	Réalisation du dossier, codage et tests unitaires	10 heures	Mer 23/05/18	Jeu 24/05/18	31	

	Plan des Itérations	
	Docs	Justification
ITÉRATION 1	- Tâches de mise en oeuvre (TMO) pour configurer les émetteurs/récepteurs avec le logiciel Rf Magic. - TMO pour tester les E/R avec 2 machines Windows avec RealTerm et prise de connaissance sur l'état du RTS. - Réalisation d'un programme test sous QtCreator permettant de faire communiquer les deux modules d'une RPI à une machine Windows. - Diagramme de classe pour la communication avec 1 à 2 méthodes qui sont fonctionnelle.	- Permet de configurer les E/M pour qu'ils communiquent correctement. - Vérification de la bonne communication entre les deux modules : Ecriture/Lecture. - Communication sans logiciel particulier qui peut donc s'adapter à n'importe quel système d'exploitation (Windows, Linux ou Mac OS X). - Montrer le début du raisonnement grâce aux méthodes fonctionnelles présentées et ainsi avoir une vision globale sur la partie à réaliser.
ITÉRATION 2	- Continuation de la classe communiquer et ajout de la classe qui gère le protocole DATA. - Diagramme de classe complété avec explications des méthodes ajoutées (2-3). - Mise en oeuvre sur la Raspberry réalisée par les EC. - Mise en accord avec les autres IR sur l'interface graphique à réaliser.	- Assurer la bonne communication des données concernant le protocole DATA. - Le diagramme de classe servirai de vue globale de la partie communication. - Test sur la Raspberry des EC pour une première démonstration. - Permet d'éviter un confusion générale lors de la mise en oeuvre de l'interface graphique finale.
ITÉRATION 3	- Finalisation des classes de communication et intégration dans le programme final. - Démonstration fonctionnelle finale du projet. - Diagramme de classe final.	- Appuyer la revue avec le diagramme de classe final afin d'expliquer le but des programme de manière globale et précise. - Assurer une communication complète des données entre la station au sol et le drone.

Pour commencer, j'ai réalisé un plan des itérations regroupant les différentes tâches que je dois effectuer dans un ordre plus approprié, afin de comprendre de manière plus globale le fonctionnement de la communication qui sera équipée sur le drone.

Commencement des tâches :

Dans un premier temps, j'ai dû configurer les émetteurs/récepteurs qui serviront de communication entre l'EDD et la GCS. En effet, ces émetteurs/récepteurs émettent en ondes radio à une fréquence de 433 MHz.
Ils disposent donc d'une portée assez importante ce qui était idéal pour le projet.
Pour les configurer j'ai utilisé un logiciel appelé « RF-Magic » :

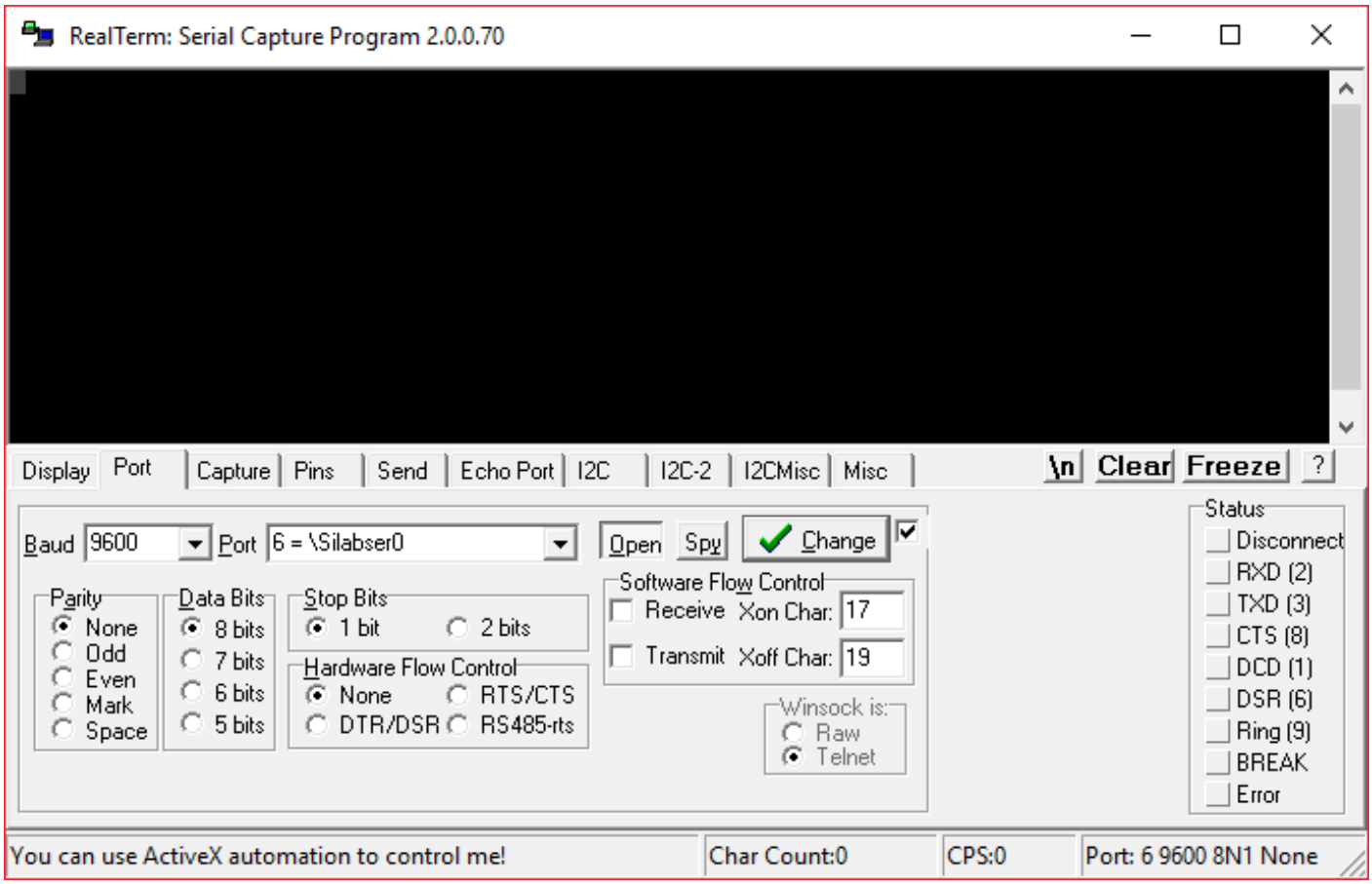


Après les avoir configuré il fallait les tester pour deux raisons :

- Vérifier qu'ils communiquaient bien entre eux à partir de deux machines différentes.
- Comprendre leur fonctionnement pour trouver une future solution.

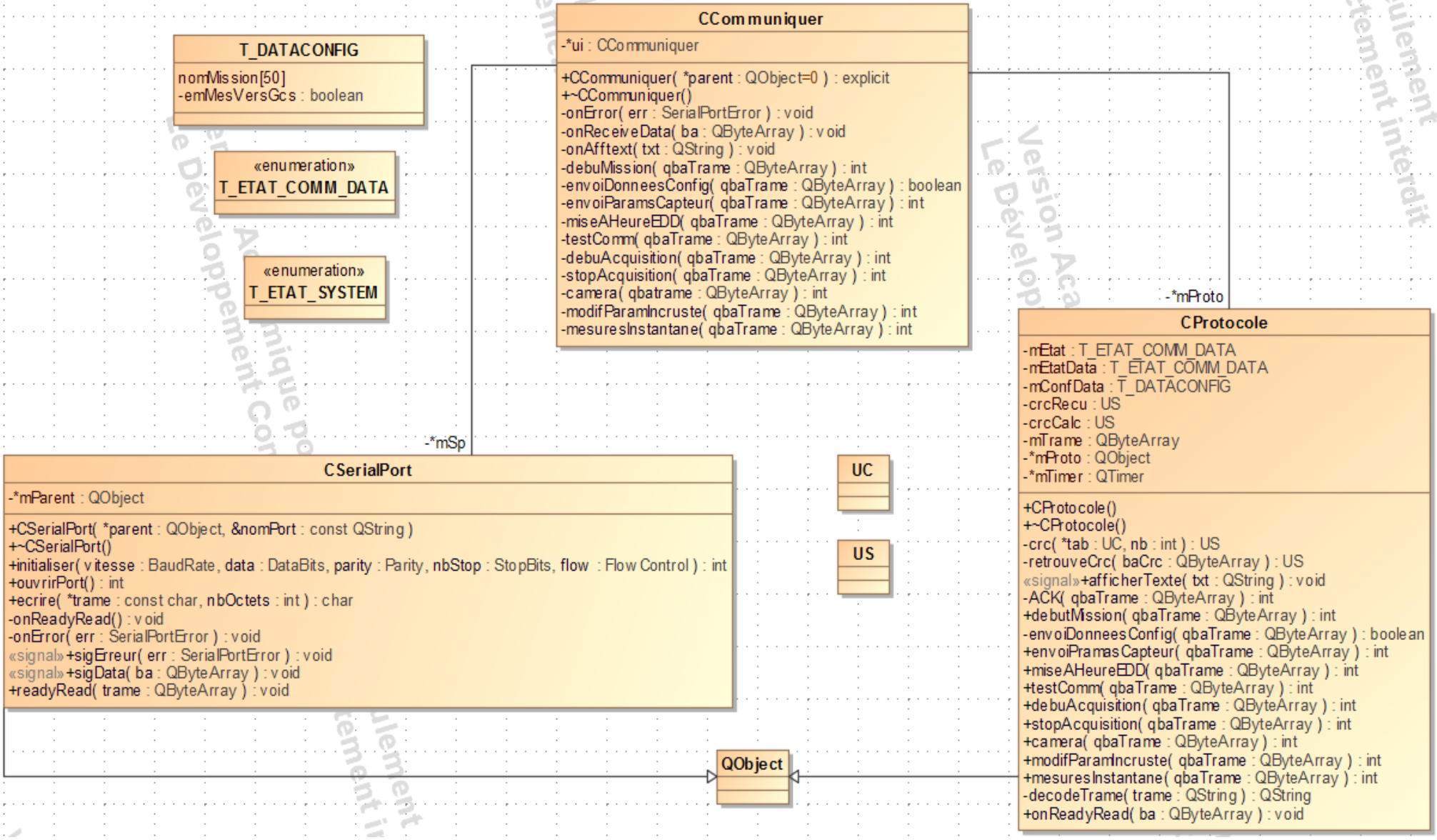
Dans un premier temps, j'ai demandé à l'un de mes partenaires d'installer « Realterm » qui est un logiciel permettant d'observer l'émission et la réception de données telles que des phrases tapées manuellement.

Exemple :



Une fois que le fonctionnement était acquis et la communication vérifiée, je me suis penché sur une tâche de mise en oeuvre qui n'est autre qu'un simple programme Qt en C++ qui permet d'émettre ou de recevoir à partir de n'importes quels systèmes d'exploitation, que ce soit Linux, Windows, ou Mac OS X.

Je l'ai donc testée à partir d'une carte Raspberry vers une machine sous Windows ou Mac OS X en branchant les émetteurs/récepteurs de chaques côtés.


T_DATACONFIG
nomMission[50]
-emMesVersGcs : boolean
«enumeration»
T_ETAT_COMM_DATA
«enumeration»
T_ETAT_SYSTEM
CCommuniquer
-*ui : CCommuniquer
+CCommuniquer(*parent : QObject=0) : explicit
+~CCommuniquer()
-onError(err : SerialPortError) : void
-onReceiveData(ba : QByteArray) : void
-onAfftext(txt : QString) : void
-debuMission(qbaTrame : QByteArray) : int
-envoiDonneesConfig(qbaTrame : QByteArray) : boolean
-envoiParamsCapteur(qbaTrame : QByteArray) : int
-miseAHeureEDD(qbaTrame : QByteArray) : int
-testComm(qbaTrame : QByteArray) : int
-debuAcquisition(qbaTrame : QByteArray) : int
-stopAcquisition(qbaTrame : QByteArray) : int
-camera(qbatrame : QByteArray) : int
-modifParamIncruste(qbaTrame : QByteArray) : int
-mesuresInstantane(qbaTrame : QByteArray) : int
-*mProto
CProtocole
-mEtat : T_ETAT_COMM_DATA
-mEtatData : T_ETAT_COMM_DATA
-mConfData : T_DATACONFIG
-crcRecu : US
-crcCalc : US
-mTrame : QByteArray
-*mProto : QObject
-*mTimer : QTimer
+CProtocole()
+~CProtocole()
-crc(*tab : UC, nb : int) : US
-retrouveCrc(baCrc : QByteArray) : US
«signal»+afficherTexte(txt : QString) : void
-ACK(qbaTrame : QByteArray) : int
+debutMission(qbaTrame : QByteArray) : int
-envoiDonneesConfig(qbaTrame : QByteArray) : boolean
+envoiPramasCapteur(qbaTrame : QByteArray) : int
+miseAHeureEDD(qbaTrame : QByteArray) : int
+testComm(qbaTrame : QByteArray) : int
+debuAcquisition(qbaTrame : QByteArray) : int
+stopAcquisition(qbaTrame : QByteArray) : int
+camera(qbaTrame : QByteArray) : int
+modifParamIncruste(qbaTrame : QByteArray) : int
+mesuresInstantane(qbaTrame : QByteArray) : int
-decodeTrame(trame : QString) : QString
+onReadyRead(ba : QByteArray) : void
-*mSp
CSerialPort
-*mParent : QObject
+CSerialPort(*parent : QObject, &nomPort : const QString)
+~CSerialPort()
+initialiser(vitesse : BaudRate, data : DataBits, parity : Parity, nbStop : StopBits, flow : Flow Control) : int
+ouvrirPort() : int
+ecrire(*trame : const char, nbOctets : int) : char
-onReadyRead() : void
-onError(err : SerialPortError) : void
«signal»+sigErreur(err : SerialPortError) : void
«signal»+sigData(ba : QByteArray) : void
+readyRead(trame : QByteArray) : void
UC
US
QObject

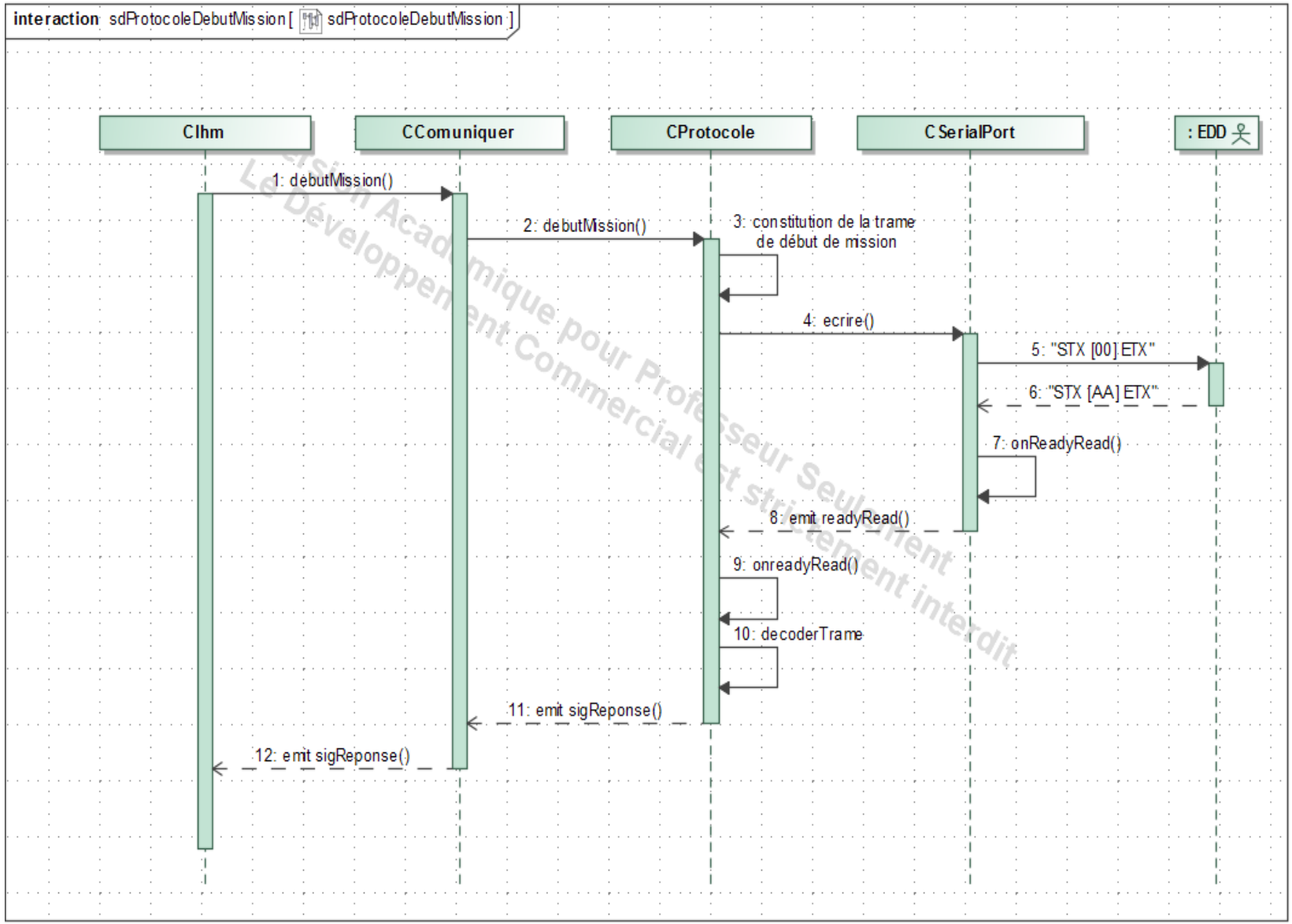
Les classes CCommuniquer, CProtocole et CSerialPort ont des relations particulières.

En effet, la classe CProtocole est composée d'un objet de la classe CSerialPort. Dans le cas du diagramme de classe ci-dessus l'objet CSerialPort est « mSp ». De plus la classe CCommuniquer est composé d'un objet de la classe CProtocole.
Cependant la classe CCommuniquer joue le rôle de « classe d'interface », c'est-à-dire qu'elle s'occupe uniquement de réaliser un relais entre la classe CProtocole et l'interface graphique.

Sur la page suivante un diagramme de séquence qui schématise mieux le prince de requête et de réponse pour l'ordre de début de la mission par exemple.

Dans le cas suivant, la GCS envoie une requête contenant l'ordre de début mission. Cet requête est traduit par l'appel de la méthode debutMission() de la classe CProtocole.

Une fois dans cette classe la trame de l'ordre de début de la mission est formée et envoyée sur le port série afin de la communiquer au drone.
Celui-ci émet donc des signaux comme réponse permettant à la GCS de vérifier que l'ordre a bien été reçu et bien traité.



Les modules émetteurs/récepteurs :

Les modules permettant la communication entre la GCS et l'EDD émettent des ondes radio.

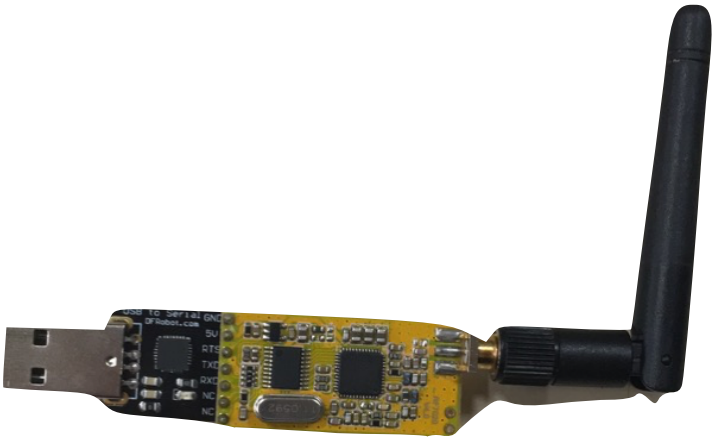
Les ondes radio sont des ondes électromagnétique pouvant avoir une fréquence comprise entre 9 kHz et 300 GHz.

Elles se propagent dans l'espace libre : autour de la Terre par exemple.

Pour finir sur les ondes radio il faut savoir qu'elles doivent être modulées afin d'assurer une bonne communication.

En ce qui concerne les émetteurs/récepteurs ils ont différents paramètres par défaut :

- Leur fréquence = 434 MHz ce qui correspond à la plage de fréquence des ondes radio,
- Une vitesse de transmission de 9600 bps qui est identique pour la modulation, qui est une modulation GFSK(Gaussian Fréquence Shift Keying),
- Une puissance de sortie de 500 mW maximum,
- Une intervalle de fréquence de 200 kHz qui permet la communication dans plusieurs canaux en cas d'émission multiples,
- Une tension supportée comprise entre 4,7 V et 8 V.



La modulation GFSK(Gaussian Frequency Shift Keying) est tirée d'une modulation FSK(Frequancy SHift Keying) dite modulation de fréquence.

Dans la FSK, le signal modulé varie entre des fréquences prédéterminées. En effet, par exemple pour un 0 une fréquence sera attribuée et pour un 1 une fréquence différente sera attribuée.

La différence entre une modulation GFSK et FSK est minime puisque dans une modulation GFSK un filtre dit gaussien qui permet de lisser les courbes de fréquences, d'où le « Gaussian » de GFSK.

Le Protocole DATA :

La communication des données se fait à l'aide de trame d'une certaine forme qui est définie par la documentation du protocole DATA. En effet, par exemple pour informer le drone du début de la mission, la trame suivante est créée :

0x02 [00] 0x03

Il existe donc deux états, Avant mission et Pendant mission, qui permettent au drone d'effectuer certains ordre ou non qui lui sont demandés.

Voici un extrait de la documentation qui permet d'obtenir les informations nécessaires pour réaliser la trame correspondante :

Fonction	syntaxe	sens	commentaire
Départ mission	**[00]**	GCS→EDD	Informe le drone du départ de la mission. La GCS doit émettre ce message une fois les données de configuration et d'incrustation envoyées. L'état du système passe à « **pendant mission** ».

Le protocole étant en cours de développement je n'ai à l'heure actuelle qu'une méthode correcte permettant de construire une trame :

```
int CProtocole::debutMission()
{
    QByteArray qbaTrame;
    if(mEtat != PENDANT_MISSION)
    {
        mEtat = PENDANT_MISSION;
        qbaTrame = "\x02[00]\x03";
        mSp->ecrire(qbaTrame,6);
        emit afficherTexte(QString(qbaTrame));
    }
    return 0;
}
```

Construction de la trame

Ecriture de la trame sur le port série

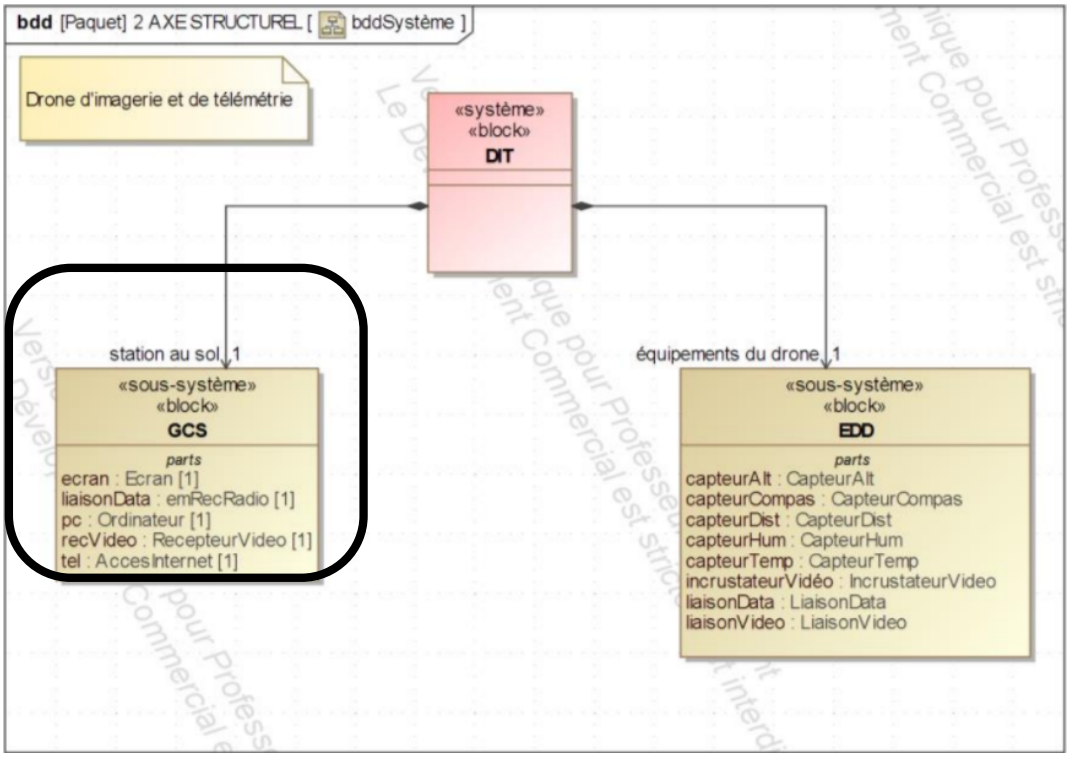
IR 2 – Supervision et Historisation

Présentation Personnelle :

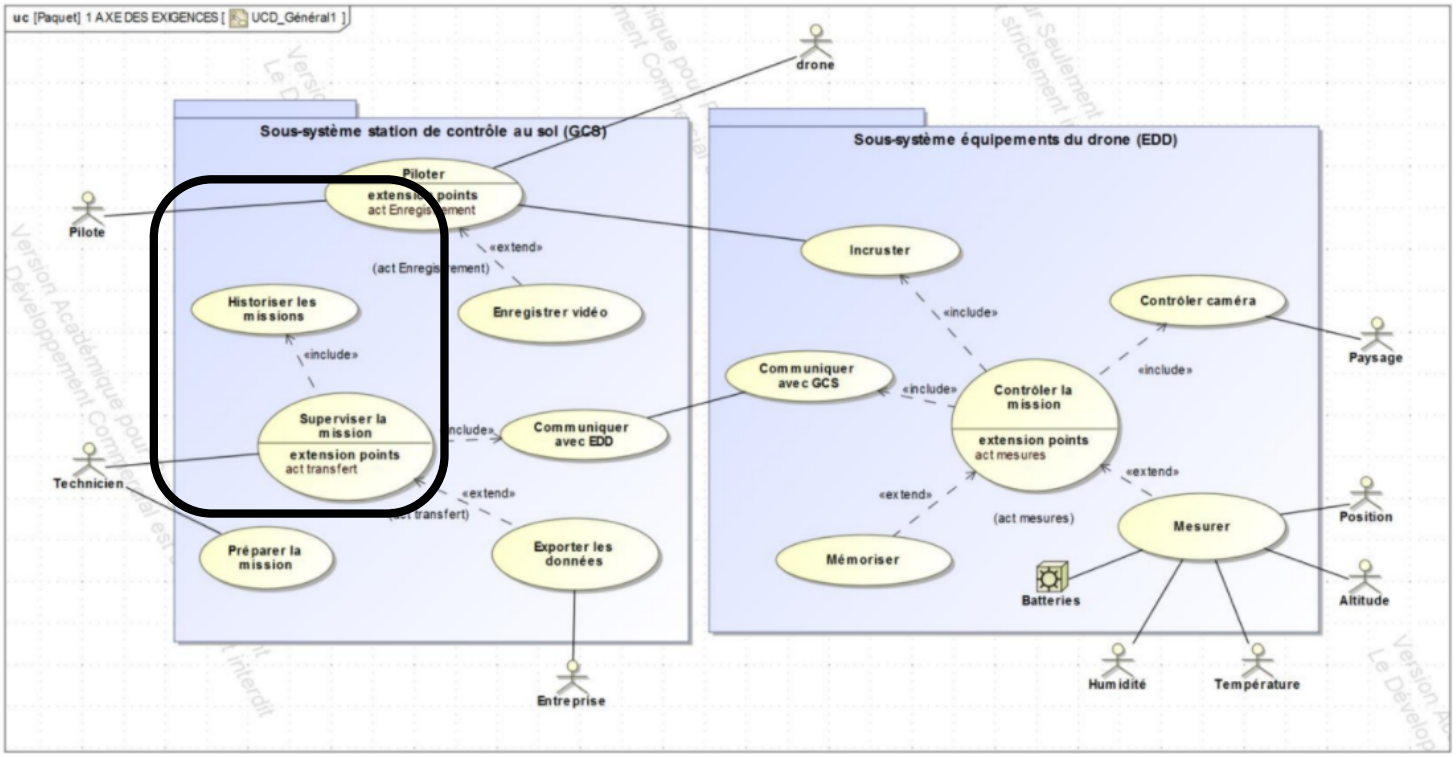
Dans le cadre de ma formation en BTS SN IR au lycée Alphonse Benoît, durant ma seconde année j'ai du réaliser un projet industriel me permettant de mettre en pratique mes connaissance théorique

Ce projet technique, a été réalisé en collaboration avec deux autres étudiants de la section IR (DEMONCHY Maxime, CARRILLO Thomas), ainsi que quatre étudiants de la section EC (BUFFE Jordan, LOZANO Dorian, TISON Sylvain, BOTTIGLIERO Nicolas). Celui-ci consiste a surveiller l'état d'édifice et d'ouvrage, pour cela la solution est d'utiliser un drone afin d'effectuer différentes mesures telles que la température, ou l'humidité et de filmer pour voir les défaillances éventuelle et pouvoir intervenir rapidement

En ce qui me concerne, j'ai été chargé de rendre possible la préparation, la supervision, et l'arrêt des missions grâce à une IHM, ainsi de que pouvoir faire une historisation et un stockage, des missions et des mesures de celle-ci depuis la station au sol (voir diagramme ci-dessous)

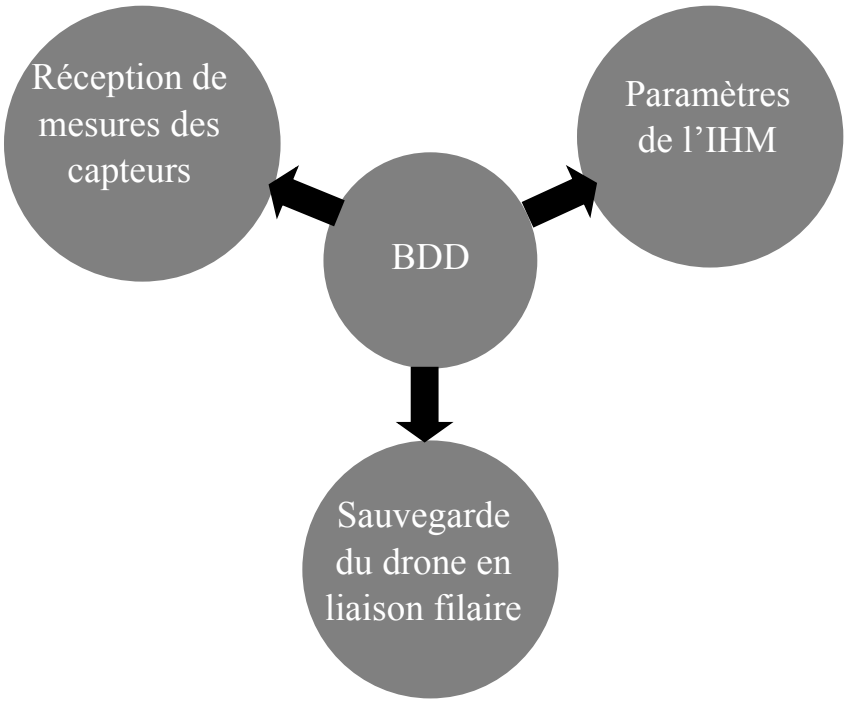


Ma partie concerne les blocs encadrés dans le diagramme de cas d'utilisation suivant :



Mon but est donc de créer et coder une application pour contrôler et superviser l'acquisition des mesures, ainsi que de mettre en place une base de données qui permettra de gérer l'historisation des missions

Voici un schéma simple qui explique les différents rôles de la base de données :



Au début du projet, j'ai élaboré une planification prévisionnelle des tâches à accomplir pour mener à bien le projet en faisant un tableau d'itérations :

Désignation	Itération	Justification
Conception de L'IHM **Conception de la BDD :** ➔ **Conception MCD** ➔ **Conception MLD** ➔ **Codage BDD** **Conception CBDD :** ➔ **Connexion** ➔ **Sauvegarder MESURE**	1	Après avoir lu le cahier des charges la conception de l'IHM me paraissait important pour voir l'application générale utiliser et voir les besoins de la BDD Ainsi la conception du MCD et MLD ma permis de pouvoir savoir les méthodes utiliser dans la classe CBDD
Conception Finale de CBDD **Conception Finale l'IHM**	2	Une fois la BDD fini il me fallait terminer le CBDD pour pouvoir l'intégrer à l'IHM Finale
Conception Finale de l'IHM **Intégration de CCommuniquer**	3	Une fois le CBDD fini il faut finir maintenant l'IHM pour être prêt a intégrer le CCommuniquer

Dans cette planification n'ont pas été comptées les réunions, car celle-ci n'étais pas prévues, ni les problèmes rencontrés, malgré que ça ait eu de l'influence sur le temps de travail.

Travail réalisé

Présentation du programme de la station au sol (GCS)

Ma tâche principale pour ce projet, est de coder une interface homme/machine (IHM) permettant de préparer et superviser les missions, mais aussi en arrière-plan de communiquer avec la base de données afin de permettre l'historisation des mesures

J'ai programmé cette interface en C++, sous QT (Sous Windows). Il s'agit donc de programmation objet

Le programme ayant plusieurs fonctions différentes, il fonctionnera avec une classe principale (IHM), une classe de gestion de la base de données(CBDD), et une classe gérant la communication avec le module qui est fait par mon camarade CARRILLO Thomas

Présentation de l'interface utilisateur

L'interface homme/machine, c'est la partie « visuel » de l'application, la partie qui va avoir des interactions avec le technicien (afficher des données, cliqué sur un bouton…).

Le principe sous Qt , c'est de lier un signal émis par un élément de l'interface (un boutons par exemple), qui est déclenché par une action de l'utilisateur (un clic), à un slot, une méthode présente dans la classe principale.

L'idée pour cette interface, est de se baser sur trois étapes de la mission :

- L'étape **Avant mission**

  Le technicien peut saisir le nom de la mission, renseigner son nom ainsi que celui du copilote, renseigner le lieu, régler la date et l'heure du début de la mission, choisir le port série pour la liaison data ainsi que de tester la communication, choisir d'émettre les mesures en temps réel vers la station au sol ainsi que le temps de rafraîchissement des mesures, Choisir s'il veut incruster les mesure sur la vidéo ,ainsi que la date et l'heure, et pour finir sélectionner les capteur qu'il veut utiliser pendant la mission

- L'étape **Pendant mission**

  Le technicien peut démarrer et arrêter la mission, démarrer et arrêter l'acquisition des mesures, modifier les paramètres d'incrustations vidéo, activer la vidéo ou l'appareil photo de la caméra, prendre des photos, enregistrer, mettre en pause , ou arrêter la vidéo, choisir dans la liste des capteur choisir précédemment les capteurs a afficher ou non, et voir en temps les mesures en temps réel les mesures qu'il a choisit

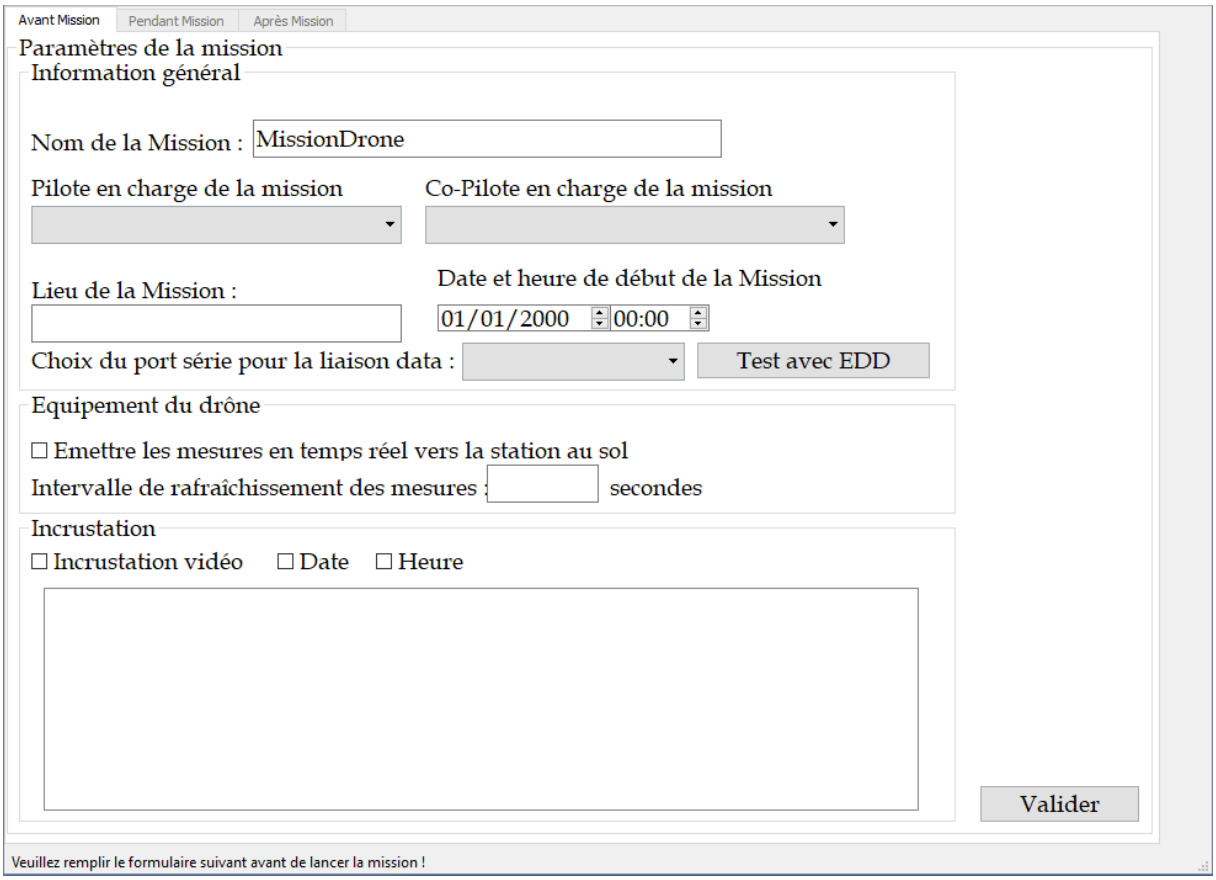
-

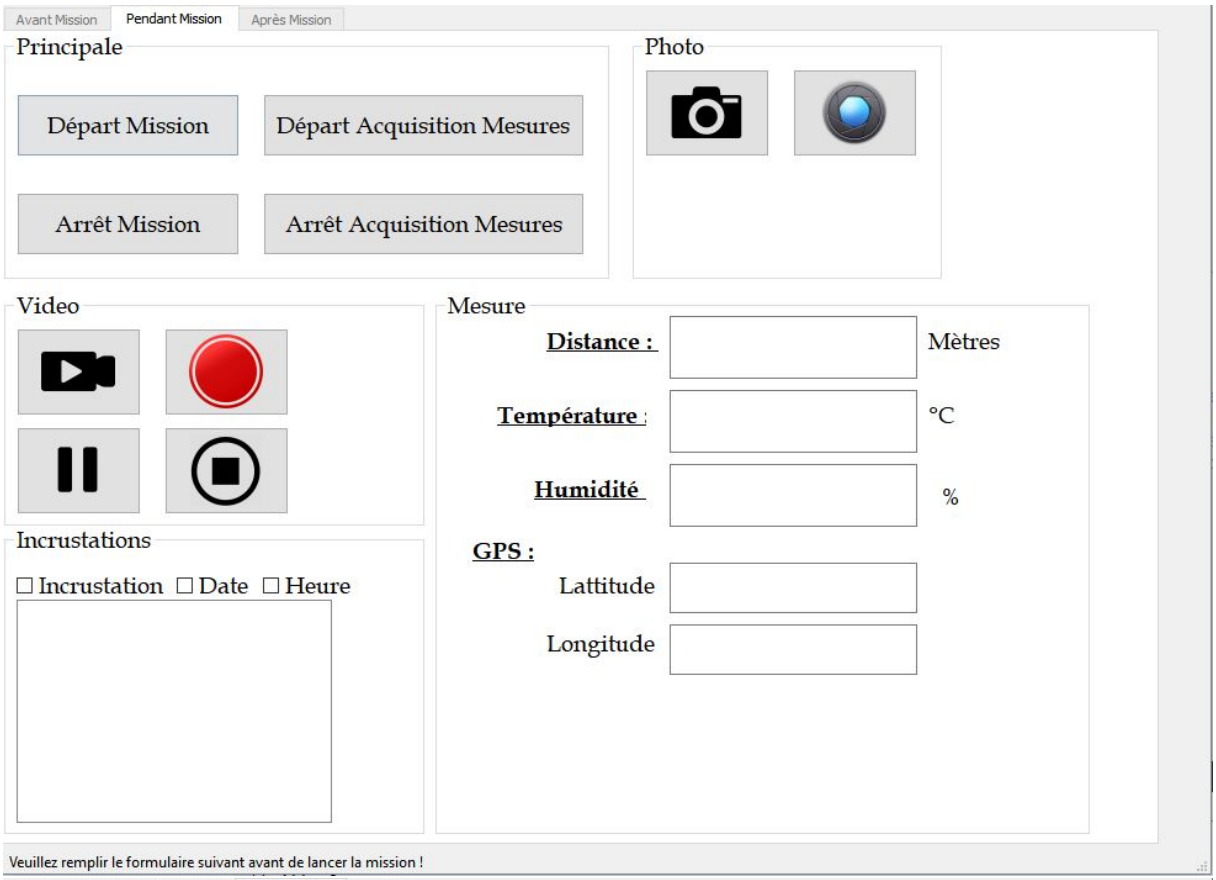
-

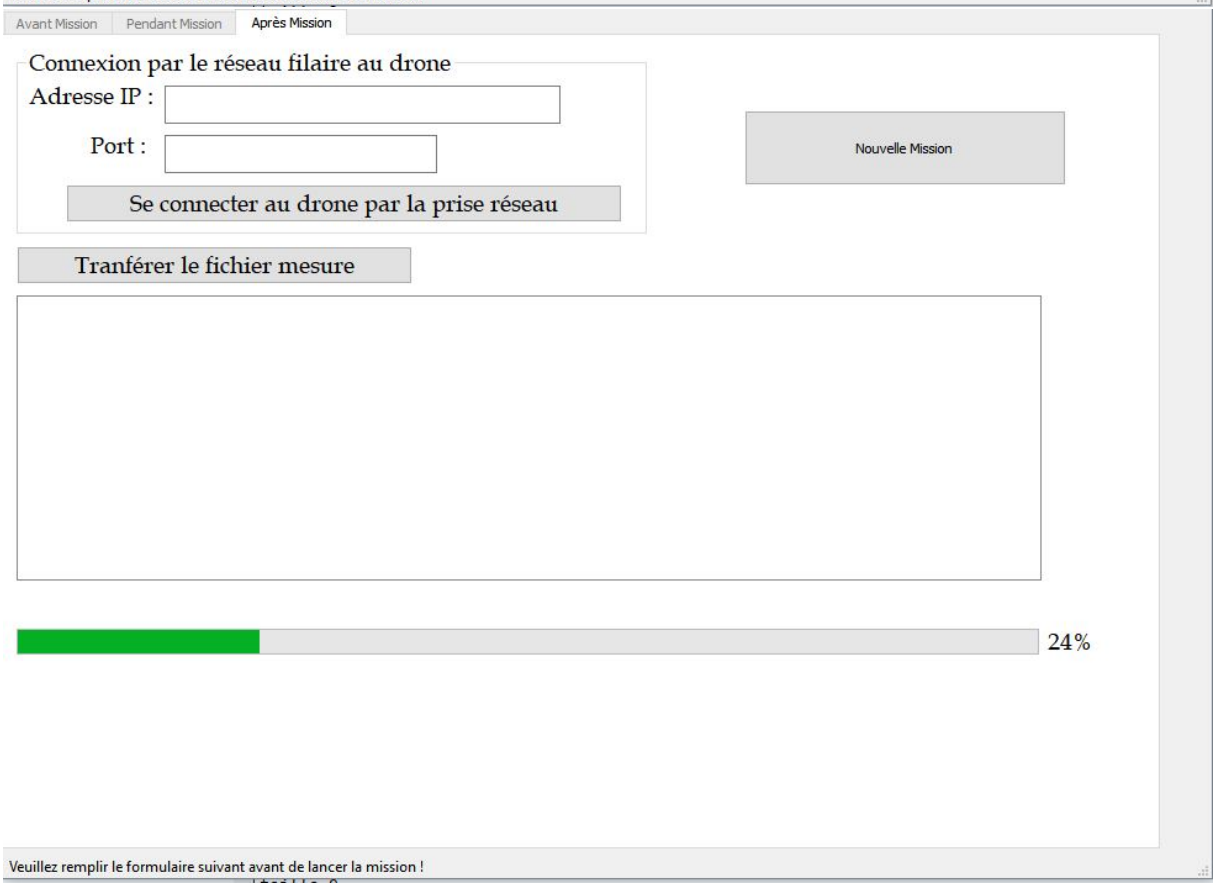
- L'étape **Après Mission**

  Le technicien peut saisir l'adresse IP, ainsi que le port pour ce connecter au drone par la prise réseau, et transférer le fichier de mesures stocker sur le drone, et commencer une nouvelles mission

J'ai donc décidé de créer trois zones sur l'interface, grâce au widget « groupBox ». Cela me permet aussi de les activer/désactiver lors d'un passage d'une étape a l'autre afin d'éviter une mauvaise manipulation de l'utilisateur

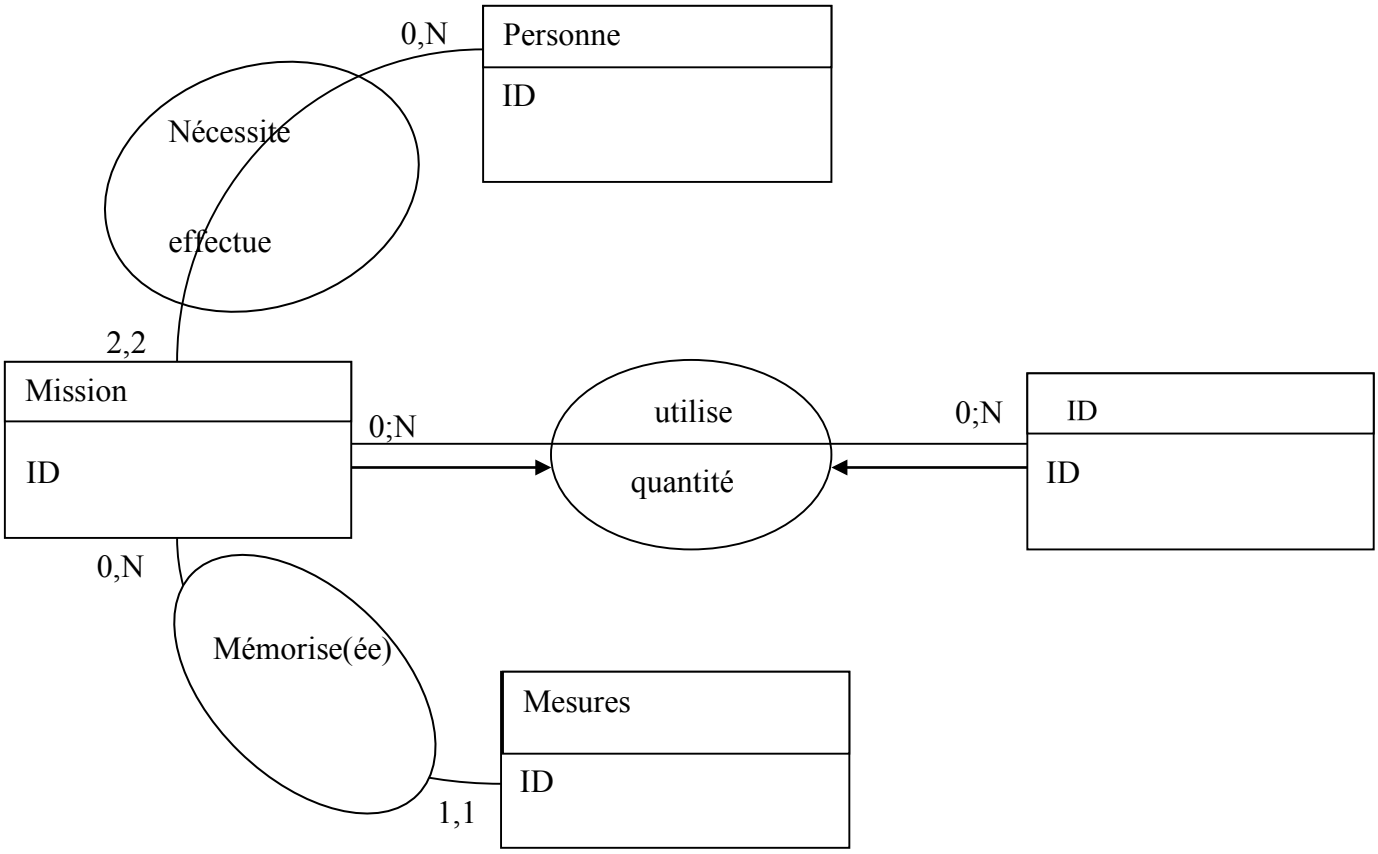



Avant Mission
Pendant Mission
Après Mission
Principale
Départ Mission
Départ Acquisition Mesures
Arrêt Mission
Arrêt Acquisition Mesures
Photo
Video
Mesure
Distance :
Mètres
Température :
°C
Humidité
%
GPS :
Lattitude
Longitude
Incrustations
Incrustation
Date
Heure
Veuillez remplir le formulaire suivant avant de lancer la mission !


Avant Mission
Pendant Mission
Après Mission
Connexion par le réseau filaire au drone
Adresse IP :
Port :
Se connecter au drone par la prise réseau
Nouvelle Mission
Tranférer le fichier mesure
24%
Veuillez remplir le formulaire suivant avant de lancer la mission !

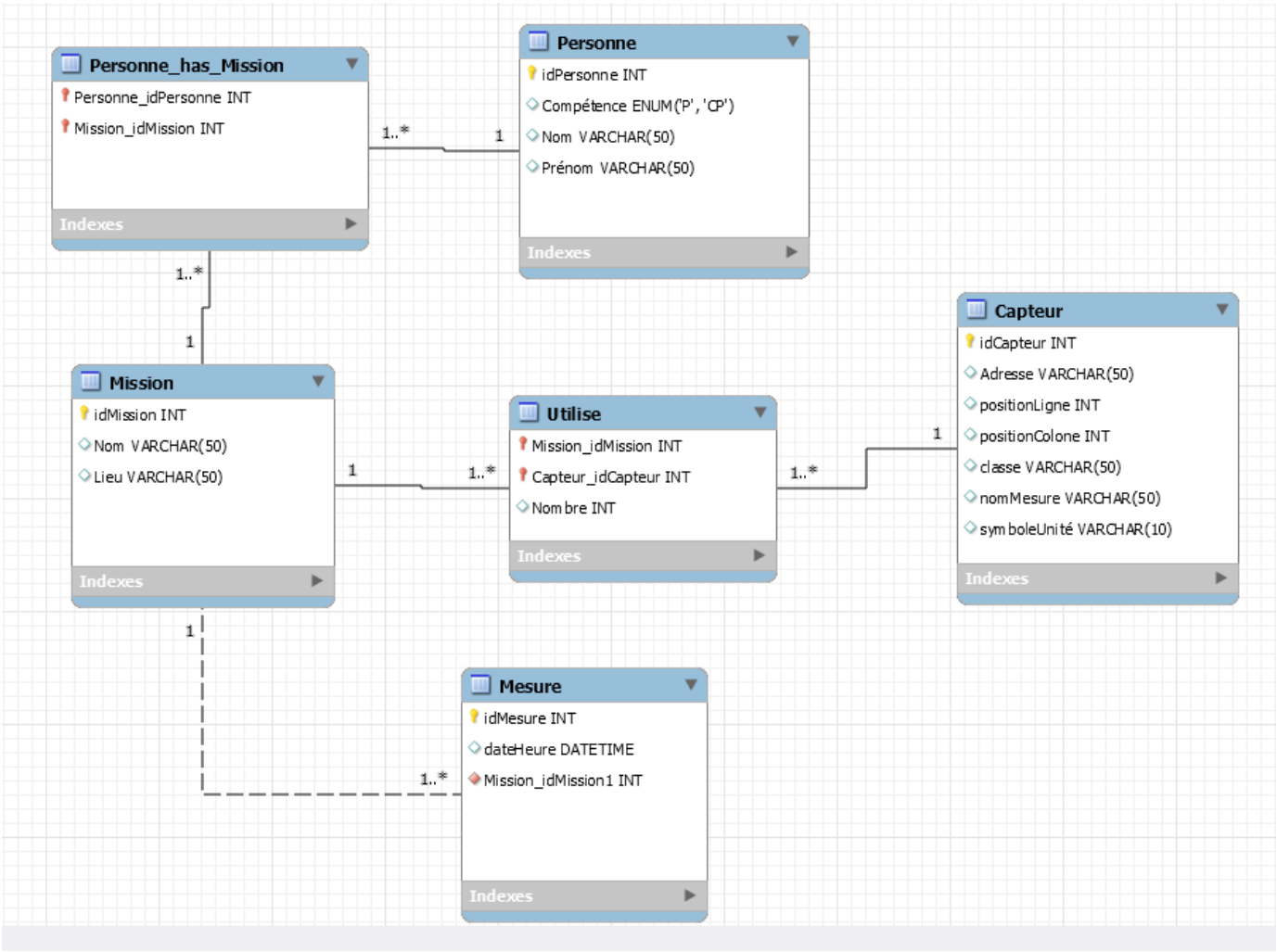
Présentation de la base de données (BDD)

Avant de créer la base de données, j'ai du réfléchir au classe que dont j'allais avoir besoin dans celle-ci. Pour cela j'ai réalisé un modèle conceptuel des données (**MCD**) qui a pour but de représenter de façon structurée les données qui seront utilisées par le système d'information.



Le MCD m'a ensuite permis de pourvoir réaliser un Modèle Logique de Données (**MLD**) qui est la modélisation logique des données qui tient compte du niveau organisationnel des données. Il s'agit d'une vue logique en terme d'organisation de données nécessaire à un traitement. En symbolique, il y a des flèches entre les tables au lieu des patates entre les entités du MCD

Le MLD pour la base de données que je doit créer est le suivant :



Une des tâches de ma partie est d'assurer l'historisation des missions. Cela comprend, l'enregistrement de chaque, avec sa configuration, et de toute les mesures faites durant celle-ci.

Cela permet d'analyser les données après la mission, ou de comparer à des missions antérieures.

Les données a stocker peuvent être représentée sous formes de différents tableaux, selon le types de données

Exemples :

- Enregistrement des missions

ID Mission	Date	Heure de début	Durées	Nom	Lieu	Pilote Copilote
1	26/11/2017	13 :45	02 :50	Mission01	Pont du Gard	Busseret Demonchy
2	13/12/2017	16 :30	01 :05	Mission01	Viaduc de Millau	Busseret Demonchy
3	06/01/2018	9 :22	00 :42	Mission01	Pont du Gard	Busseret Demonchy

- Enregistrements des mesures

ID Mesure	ID Mission	Date / Heure	Nom Capteur	Mesure
1	1	26/11/2017 13:55	capttemp	24
2	1	26/11/2017 13 :55	captdist	11
3	2	13/12/2017 16 :46	captdist	4

- Enregistrements des capteurs

ID Capteur	Nom	Type	Symbole
1	capttemp	Température	°C
2	captdist	Distance	m
3	captalt	Altitude	m

Voici comment j'ai décidé de modéliser la base de données de la station au sol :

La tables « Personne » contient un enregistrement de toute les personnes qui pourront s'occuper de la station au sol entre les pilote du drone et les copilote elle contiendra l'identifiant de chaque personne, leur compétence suivi de leur nom de famille et leur prénom

La tables « Mission » contient un enregistrement de chaque mission avec son Identifiant (id) qui sera auto incrémenter par le système de gestion, le nom de la mission ainsi que le lieu

La table « Mesures » contiendra un enregistrement de chaque mesures (une par capteur).Ce sera donc la table qui contiendra le plus de données. Les colonnes de cette table sont les numéros de la mesure auto-incrémentée, l'identifiant de la mission correspondant, la date et l'heure de la mesure, le nom du capteur associé et la mesure.

La tables « capteur » qui répertoriera chaque capteur pouvant être utilisé sur le drone ils comporteront le nom donné par le fabricant l'adresse I2C ou GPIO sur le drone, le type de capteur (température, altitude…) et le symbole et l'unité de ses mesures. Une colonne « activation » sera utilisée pour gérer l'utilisation des capteurs sur le drone lors de chaque mission

Une tables de jointure a été ajoutée automatiquement lors de la création de ce modèles entre la table « mission » et la tables « capteur », car une mission utilisera un ou plusieurs capteur, et un capteur sera utiliser par une ou plusieurs mission

De même une tables de jointure a été ajoutée automatiquement lors de la création de ce modèles entre la table « mission » et la tables « Personne », car une personne réalisera une ou plusieurs mission, et une missions sera utiliser par une ou plusieurs personnes.

Communication avec la base de données

La classe principale IHM va appeler des méthodes de l'objet créé a partir de la classe « cbdd », qui va ensuite émettre les requêtes SQL en local correspondantes.

Les méthodes de « cbdd » utilisées par l'IHM sont :

- Connexion à la BDD

Cette méthode sert a se connecter a la base de données avec les identifiants utilisateur grâce la librairie QSqlDatabase de QT

Les identifiants et paramètres sont entrés en « dur » dans le code, et donc non configurable par l'utilisateur

- Lecture de la liste des capteurs

Récupérations des noms des capteurs pour l'affichage sur l'interface afin de permettre l'activation et la sélection par le technicien. Elle contient une simple requêtes Sql « SELECT » qui liste les noms des capteurs dans la tables « capteurs », et elle la renvoie par l'intermédiaires du signal « sendListCapt »

- Lecture de la liste des pilotes et copilotes

Récupérations des noms des personnes qui peuvent être en charge de la mission pour l'affichage sur l'interface afin de permettre l'activation et la sélection par le technicien. Elle contient une simple requêtes Sql « SELECT » qui liste les noms des personnes dans la tables « personnes », et elle la renvoie par l'intermédiaires du signal « sendListPers »

- Enregistrement de la configuration de la mission

Envoie des différents paramètres choisie par le technicien pour la mission, notamment le nom de mission (obligatoire) , l'heure et la date (automatique par défaut), la liste des capteurs utilisés (tous activés par défaut, le temps d'actualisation des mesures (1 seconde par défaut) et les nom des pilotes et copilote

Un nouvel enregistrement est créé dans la table « mission » grâce à la commande « INSERT »

- Activation des capteurs

Modification d'une valeur dans l'enregistrement de chaque capteur (avec 0 : non utilisé et 1 : utilisé)

Une boucle modifie une par une les valeurs en écrasant les premières, selon la demande avec la commande Sql « UPDATE »

La méthode reçois en plus de la liste des capteurs utilisés, le nombre de capteurs enregistrés dans la base de données et c'est ainsi qu'une boucle while peut fonctionner

- Enregistrement de photos et vidéos

Envoie de l'horodatage, et du numéro de chaque photo et vidéo correspondant a leur enregistrement sur la carte SD de la caméra.
L'IHM envoie l'heure de prise du cliché ou de début de la vidéo, et le numéro de mission et après réception du numéro par le drone, envoie celui-ci

- Enregistrement des mesures

Sauvegarde de chaque mesure, une par une, dans la table « mesures ».Chaque mesure correspond à un capteur donc pour chaque mission il y aura X mesures enregistrées simultanément suivant le nombre X de capteur utilisés. Cela engendre le fait que cette table aura un grand nombre de mesures

Concernant ma partie la seul partie physique dont je peux vous parler concerne la trame DATA qui définit l'ordre que j'envoie de certaine fonctionnalité de mon IHM au drone

Cette Trame DATA est une trame en Hexadécimal / ASCII

Dans les réseaux informatiques, une trame est un bloc d'information véhiculé au travers d'un support physique qui est découper en plusieurs partie comme en nous pouvons voir ci-dessous :

Chaque fonction du protocole a le format suivant :

STX[XX](DATA ; CRC)ETX

STX : code ASCII de début de trame (0x02)

[: Est le caractère de début transmis. Il permet au lecteur de vérifier qu'il s'agit d'un début de trame.

XX : Est une valeur hexa ASCII sur 2 digits désignant la fonction du protocole.

] le caractère de fin délimitant la fonction du protocole. Il permet de s'assurer de la syntaxe de la trame reçue.

DATA ; CRC : Optionnel. Données dont la taille dépend de la fonction. Le champ DATA peut être composé de plusieurs parties séparées par des points virgules.

CRC : Crc16. Valeur sur 16 bits non signée codée en ASCII. Somme de contrôle pour vérifier le bon transport de l'information. Note : Le **CRC** n'est émis que lorsque le champ **data** est présent. Cette valeur doit être recalculée par le récepteur et comparée. Le CRC ne porte que sur les caractères transmis dans la zone **data**. Il se trouve en fin de trame et a une longueur fixe de 4 octets ASCII (poids fort en premier).

ETX : code ASCII de fin de trame (0x03)

L'acquittement est systématique pour chaque fonction du protocole (voir trame d'acquittement.

Ex : Ordre émis par la GCS : **STX [00] ETX**

Acquittement par l'EDD : **STX [AA] ETX**

Remarques : Les échangent peuvent être cryptés par une clef simple 8 bits (OU EXCLUSIF).

Malheureusement je n'est pas put faire une acquisition comme j'avais prévu de faire grâce à analog discovery ou salae car le protocole data n'était pas fini dans un premier temps et que je ne suis pas arriver a faire une trame comme je voulais

Conclusion

Pour ma part j'ai mit du temps à démarré le projet car j'ai du repartir a zéros et j'ai eu un peu de mal a me lancer
Mais une fois lancée l'IHM a très vite pris vie ensuite la BDD , ensuite j'ai eu un soucis de bibliothèque dans mon « cbdd »
Malgré ces petit problèmes ce projet m'a beaucoup apporter niveau connaissance et m'a beaucoup appris en développement de projet en groupe avec différentes revues de projet permettant de mettre a jours notre avancement
Niveau de la partie physique j'ai eu beaucoup de mal a préparer celle-ci car ma partie étais plutôt axé sur une partie logiciel plutôt que matérielle
Les étapes de ma partie qui sont finies sont celle-ci :

- La base de données
- Le coté visuel de mon IHM et quelque fonctionnalité
- Les différents utilisateurs de la base de données

Les partie en cours de développement ou de réflexion sont les suivantes :

- La communication entre l'IHM et la base de données
- La communication avec le drone

Pour ce dernier cycle de développement, je prévois donc de finir les parties en cour de développement ou de réflexion

EC 1 : Température et humidité, alimentation carte d'extension

LOZANO Dorian

Je fais parti de l'équipe 1 du projet Drone.

Dans ce groupe nous sommes 4, 2 du groupe informatique et réseau et 2 du groupe Electronique et Communication.

Je suis l'étudiant EC1, je m'occupe du capteur de Température et D'humidité et je dois m'occuper de l'alimentation de la carte d'extension pour Raspberry PI qui intègres tous les breaks outs des capteurs et autres.

Étudiant	*Liste des fonctions assurées par l'étudiant*	*Installation* : - Rpi : bus I2C et librairie C/C++ BCM2835 *Mise en œuvre* : - Valider par prototypage rapide le capteur de température et d'humidité proposé. *Réalisation* : - Participer avec les autres étudiants EC à la conception d'un schéma commun intégrant tous les composants de l'EDD. Outre le capteur, la partie alimentation de l'ensemble sera liée à ce contrat. - Concevoir une carte électronique personnelle compatible avec tous les contrats EC. *Documentation* : - Bibliothèque C/C++ d'accès au capteur. - Documents de fabrication de la carte. Ces documents devront permettre une fabrication industrielle du circuit imprimé.
EC1	EDD : UC Mesurer (température et humidité) Etudier la solution 2017 concernant le capteur et l'alimentation de la carte d'extension. Concevoir/Réaliser/Tester une nouvelle carte d'extension pour Rasperry Pi intégrant tous les breakout des capteurs et autres. Développer une petite application permettant de vérifier, la bonne acquisition des mesures de température et d'humidité.	

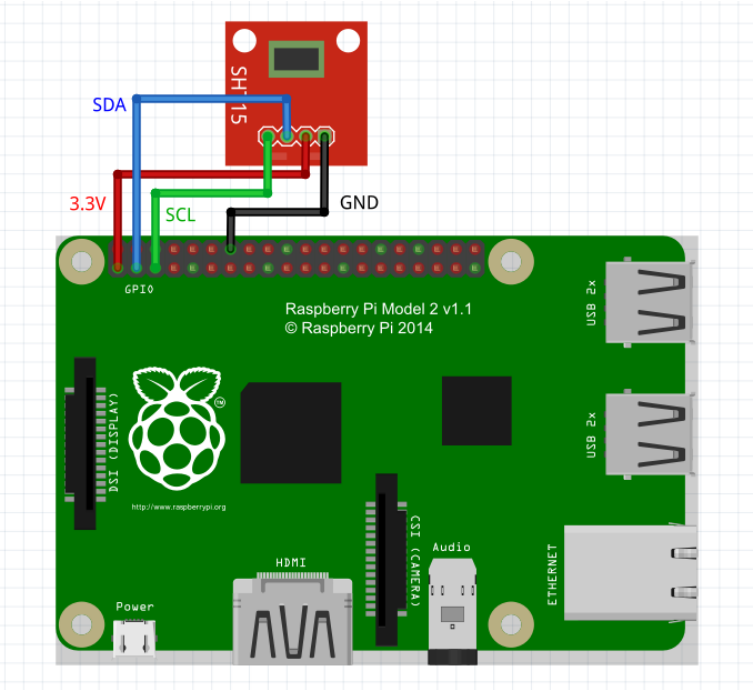
I. Choix du capteur :

Pour ce projet, je dois effectuer une récupération de la température et de l'humidité.

Pour cela, j'ai pris connaissance du module SEN-13763 proposé par mes professeurs, le capteur utilisé est le Si7021 qui permet de récupérer la température et l'humidité.

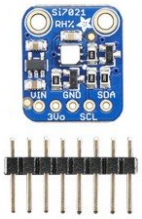
Ce capteur est alimenté en 3.3V ce que la Raspberry peut délivrer et fonctionne avec le bus I2C, la plage de mesure de ce capteur est de -40°C à 125°C et de 0 à 100% pour l'humidité, ce qui est suffisant pour l'utilisation du Drone.

Schéma de câblage rapide :



J'ai effectué quelques recherches afin de trouver d'autres capteurs pour me renseigné et ainsi voir les différents capteurs disponibles sur le marché.

J'ai choisi de faire un tableau comparatif incluant mon capteur ainsi que deux autres.

Capteur d'humidité et de T° SEN-13763	Capteur d'humidité et de T° ADA3251	Capteur de température et d'humidité Grove 101020074
		
Alimentation: 3,3 Vcc Consommation: - 300 µA pendant une mesure - 0,62 µA en veille Plage de mesure: - température: -40°C à +125°C - humidité: 0 à 100% HR Précision: - température: ±0,3 °C - humidité: ±2% HR (20 à 80% HR) Interface: I2C Dimensions: 16 x 16 mm Référence Sparkfun: SEN-13763	Alimentation: 3,3 Vcc ou 5 Vcc Plage de mesure: - température: -10°C à +85°C - humidité: 0 à 80% HR Précision: - température: ± 0,4 °C - humidité: ± 3 % HR Interface: I2C (adresse 0x40, non modifiable) Dimensions: 18 x 15 x 3 mm Référence Adafruit: 3251	Interface: compatible Grove Alimentation: 3,3 à 5 Vcc Consommation: 350 µA Plage de mesure: - température: 0°C à 70°C (±0,5°C) - humidité: 0 à 80% HR (±4,5%) Poids: 9 g Protocole: I2C Connectique non compatible avec Tinker Kit Référence Seeedstudio: 101020074
6,88 € HT **8,25 € TTC**	6,33 € HT **7,60 € TTC**	9,08 € HT **10,90 € TTC**

Pour conclure, après avoir effectué le tableau comparatif j'ai choisi de garder le capteur d'humidité et de Température SEN-13763 proposé par mes professeurs disponible sur le site Gotronic. Qui à un très bon rapport qualité prix pour sa précision et qui n'utilise pas de connectique grove pour éviter les câbles.

II. Programme du capteur sur Raspberry avec analyse de trame :

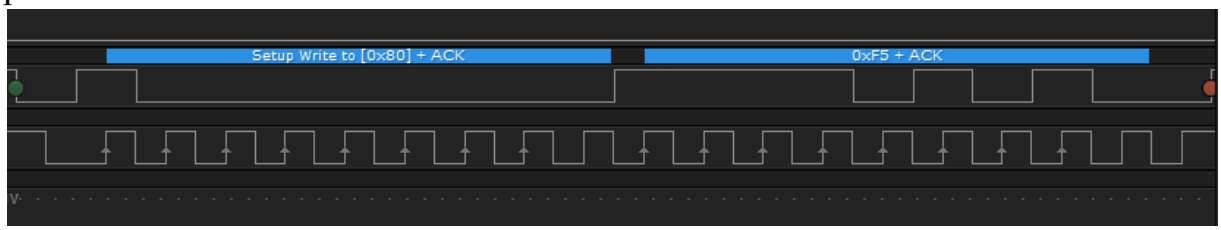
Afin de vérifier le bon fonctionnement de mon capteur j'ai cherché un programme d'exemple afin de relever les deux valeurs qui m'intéressaient, c'est-à-dire la température et l'humidité.

J'ai trouvé sur le site www.github.com un programme qui me permet de relever les deux valeurs qui m'intéressent et qui les affiches dans le terminal Raspberry, j'ai ensuite modifié le programme afin qu'il soit en boucle grâce à l'utilisation de l'instruction « While » et modifié les unités de mesures.

```
#include <sys/ioctl.h>
#include <fcntl.h>

void main()
{
while(1) {
	// Create I2C bus
	int file;
	char *bus = "/dev/i2c-1";
	if((file = open(bus, O_RDWR)) < 0)
	{
		printf("Failed to open the bus. \n");
		exit(1);
	}
	// Get I2C device, SI7021 I2C address is 0x40(64)
	ioctl(file, I2C_SLAVE, 0x40);

	// Send humidity measurement command(0xF5)
	char config[1] = {0xF5};
	write(file, config, 1);
	sleep(1);

	// Read 2 bytes of humidity data
	// humidity msb, humidity lsb
	char data[2] = {0};
	if(read(file, data, 2) != 2)
	{
		printf("Error : Input/output Error \n");
	}
	else
	{
		// Convert the data
		float humidity = (((data[0] * 256 + data[1]) * 125.0) / 65536.0) - 6;

		// Output data to screen
		printf("Humidité: %.2f RH \n", humidity);
	}

	// Send temperature measurement command(0xF3)
	config[0] = 0xF3;
	write(file, config, 1);
	sleep(1);
```

Ensuite j'ai fait des captures de trames avec Salae pour analyser le résultat.

On vient écrire à l'addresse 0x80 et on vient écrire F5 à cette adresse qui signifie prendre la mesure d'humidité.

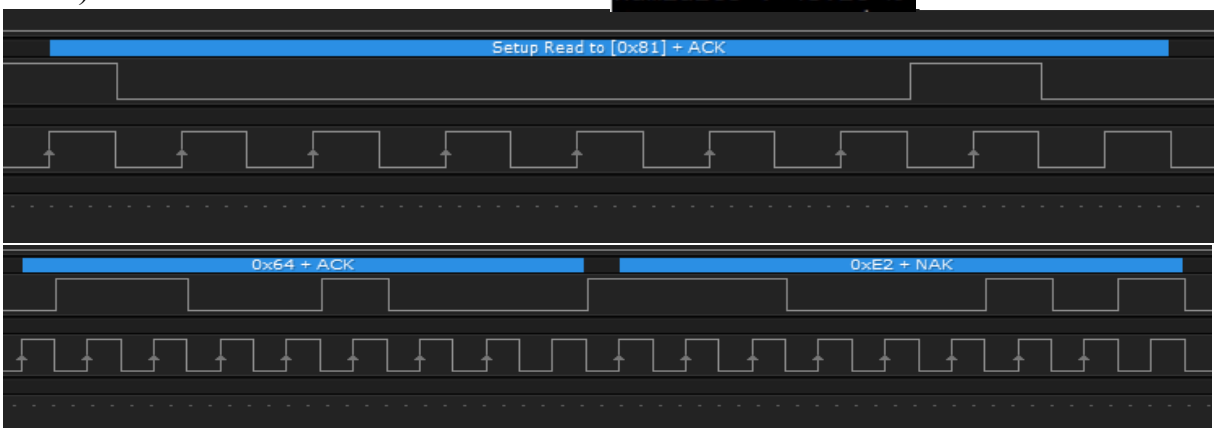


Pour lire la valeur de l'humidité ont peut voir qu'a l'adresse 0x81 ont a la valeur de l'humidité avant calcul (0x64 et 0xE2) en hexa.

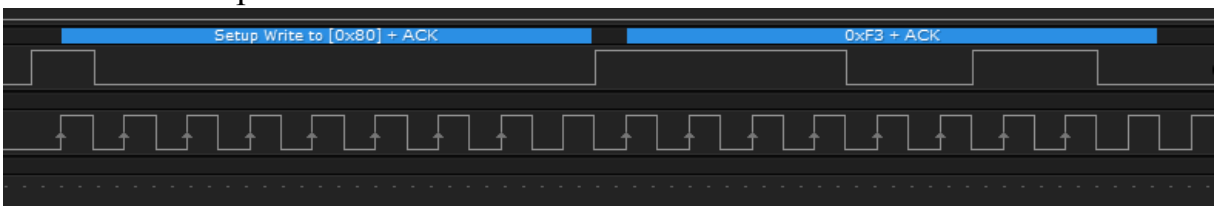
$$\%RH = \frac{125*RH\_Code}{65536} - 6$$

(Extrait de documentation)

```
Humidité : 43.26 %
```



Même procédure pour la température sauf que l'on vient écrire F3 qui signifie relevé de la température.

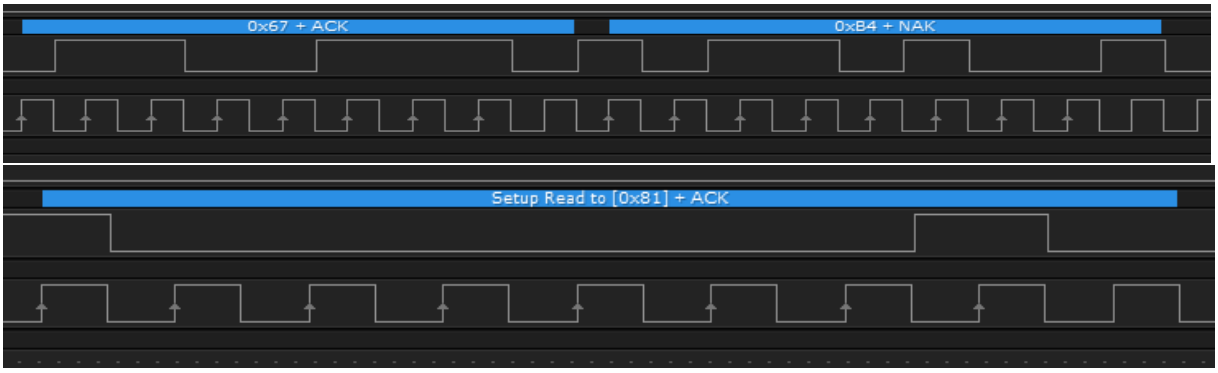


On vient relire à l'adresse 0x81 puis ont peut voir la valeur de l'humidité avant calcul (0x67 et 0xB4) en hexa.

$$Temperature\ (°C) = \frac{175.72*Temp\_Code}{65536} - 46.85$$

(Extrait de documentation)

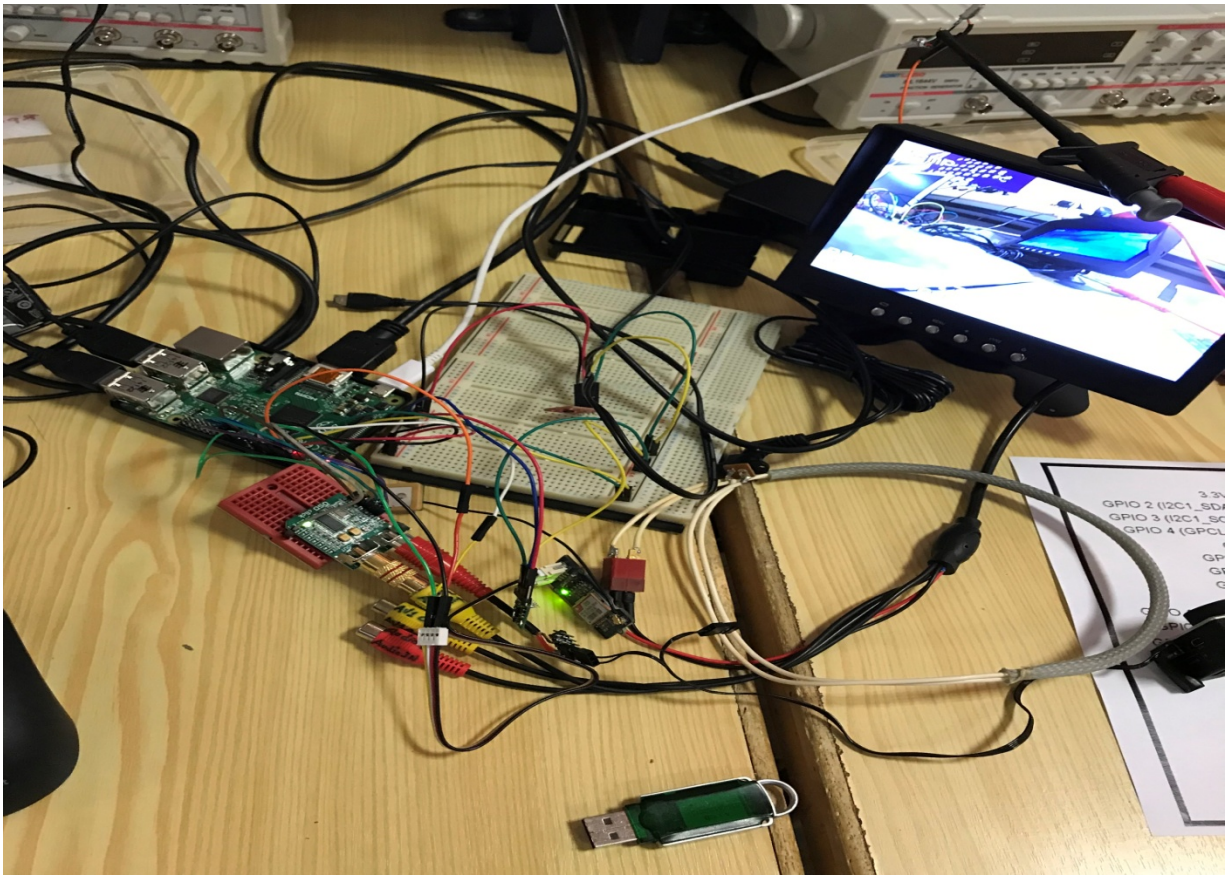
```
Temperature en degrés: 24.33 C°
```



III. Choix de l'alimentation :

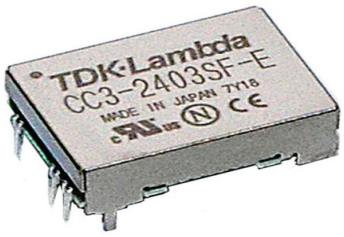
En plus de mon capteur de température et d'humidité j'ai dû m'occupés de l'alimentation de la carte d'extension et de la Raspberry implantée sur le drone, comme la Raspbery n'est plus raccordé dans les airs à une alimentation secteur j'ai choisi de prendre la source d'alimentation à partir des batteries du drone qui délivrent actuellement 7,2V en installant en amont de la Raspberry un convertisseur DC-DC pour passer de 7V à 5V ce qui conviendra à l'alimentation, ainsi qu'une intensité de 600mA.
Pour effectuer le choix du convertisseur, j'ai mesuré à l'aide d'un multimètre en mode Ampèremètre ce que consommaient tous les composants ainsi que la Raspberry, grâce à un câble USB vers micro USB que j'ai bricolé afin de pouvoir mesurer une intensité.

Démonstration que tous les capteurs fonctionnaient ensemble et au moment de la mesure mesure relevé ≈ 500mA.



L'année dernière un élève avait opté pour un convertisseur TMR 6-0511 qui permettait aussi de convertir du 7V vers du 5V et pouvait délivrer un courant de 1.2A, l'élève n'avait pas pris le temps de mesurer l'intensité globale et avait opté pour ce composant avec une forte intensité.

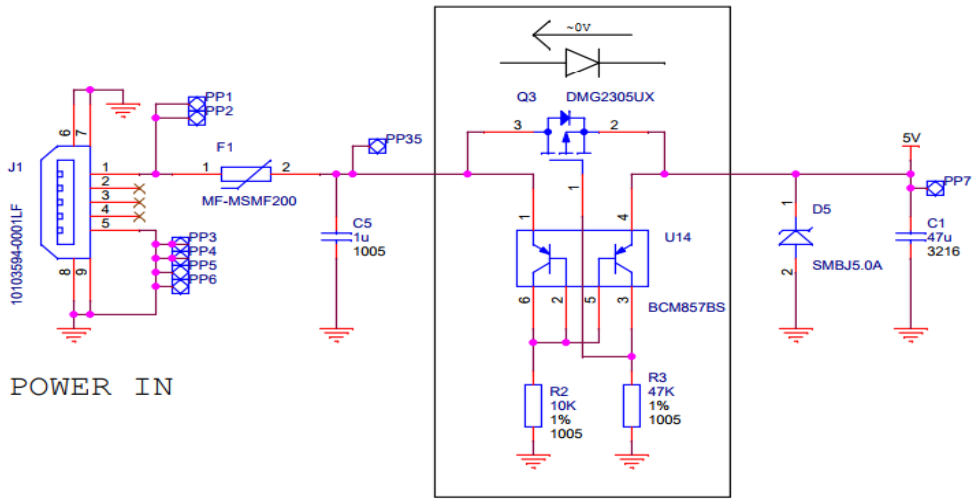
Suite à quelques recherches sur internet j'ai trouvé des composants moins chères et qui suffiraient à notre utilisation :

		
Référence fabricant : **TMR 6-0511**	Référence fabricant : **CC-3-0505SR-E**	Référence fabricant : **TMR 3-0511E**
Plage de tensions d'entrée : 4,5 → 9 V c.c	Plage de tensions d'entrée : 4,5 → 9 V c.c	Plage de tensions d'entrée : 4,5 → 9 V c.c
Courant de sortie : 1.2 A	Courant de sortie : 600mA	Courant de sortie : 600mA
Prix : 23.22 € HT 27.86 € TTC	Prix : 11.85€ HT 14.22€ TTC	Prix : 15 ,65€ HT 18.78€ TTC

J'ai choisi de prendre le TMR 3-5011E qui est tout de même moi cher et qui suffirait à alimenter tous nos capteurs ainsi que la Raspberry.

J'ai choisi d'alimenter la Raspberry directement à partir d'une broche du gpio 5V.

Dans l'image ci dessous, ont peut voir d'où arrive l'alimentation elle arrive sur le port micro USB, ensuite elle se dirige vers un fusible réarmable MF-MSMF200-2 puis arrive sur un comparateur (DMG2305UX) qui va comparer la tension de chaque côté du MOSFET, l'alimentation va passer dans la paire de transistor DMMT5401 (PI2) pour éviter de réinjecter l'alimentation dans l'usb.



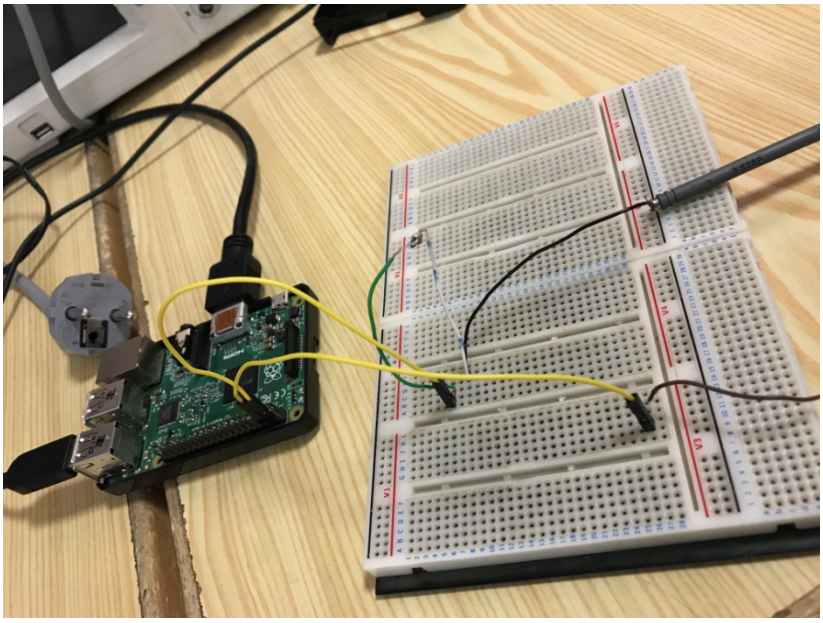
Model	Component	Value
TMR 3-05XXE	C1&C2	10µF/25V 1210 X7R
	C3	2200pF/2KV 1808 X7R
	L1	3.3µH SCD0403T/2A
TMR 3-12XXE	C1&C2	10µF/25V 1210 X7R
	C3	2200pF/2KV 1808 X7R
	L1	3.3µH SCD0403T/2A
TMR 3-24XXE	C1	10µF/50V 2220 X7R
	C3	1000pF/2KV 1808 X7R
	L1	27µH SCD0403T/0.71A
TMR 3-48XXE	C1	4.7µF/100V 2220 X7R
	C3	1000pF/2KV 1808 X7R
	L1	88µH SCD0403T/1.42A

Pour le choix du condensateur de filtrage et de l'inductance de choc on se réfère à la doc, grâce au tableau ci dessus. (Réf sur site)

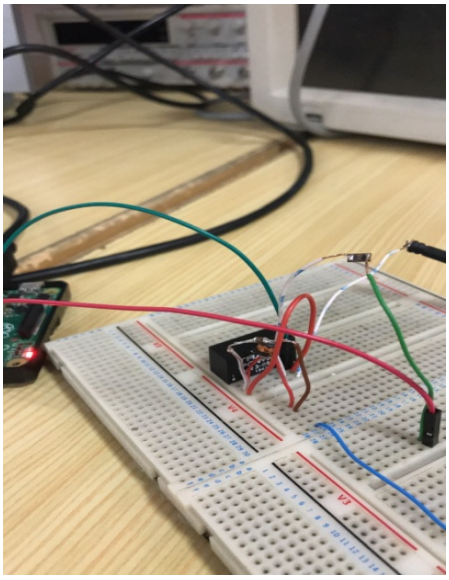
Ces deux composants permettent d'éviter les grosses variations de courants et de lisser le signal.

IV. Test alimentation raspberry par broche 5V et avec le convertisseur :

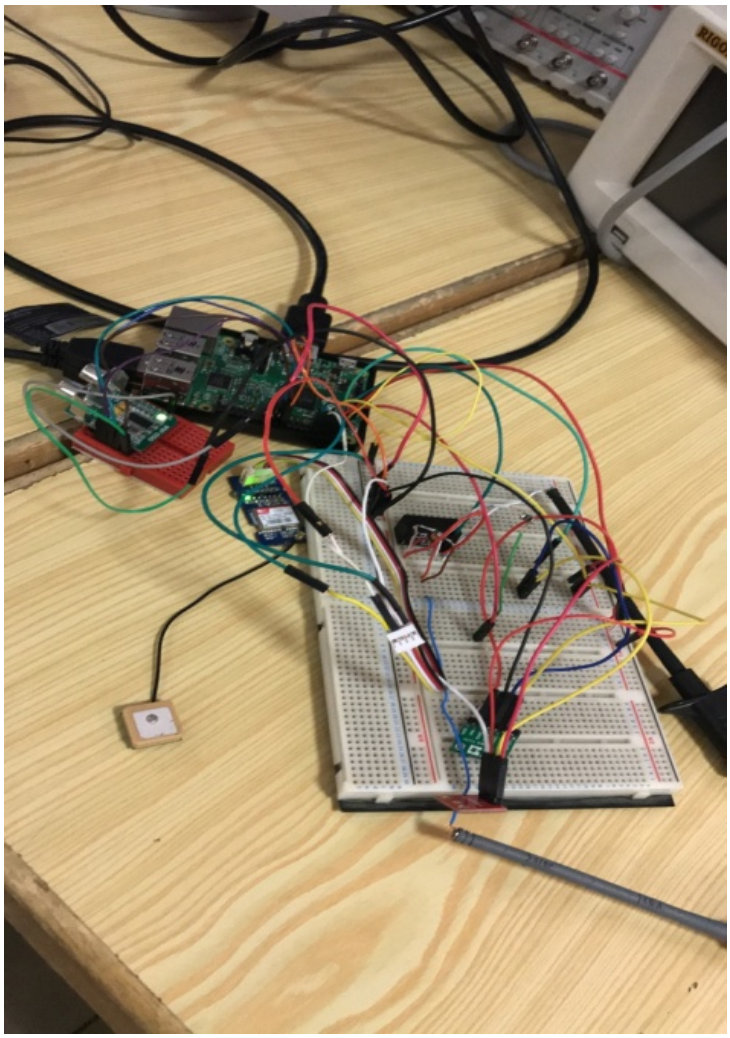
Avant de commander le nouveau convertisseur j'ai testé à l'aide d'une alimentation externe régler sur 5V et je me suis branché sur une broche 5V du GPIO du Raspberry à l'aide d'une bread board en mettant en amont de la Raspberry le fusible réarmable, tout fonctionnait bien j'ai donc commandé le convertisseur.



Une fois le convertisseur reçu j'ai testé son bon fonctionnement. Avec une alimentation externe réglée à 7.2 V, 600mA et à l'aide d'une breadboard, nous avons câblé le convertisseur avec son condensateur de 10µF et l'inductance de choc (25V 1210 X7R) ainsi que le fusible réarmable pour respecter le schéma d'alimentation de la Raspberry.



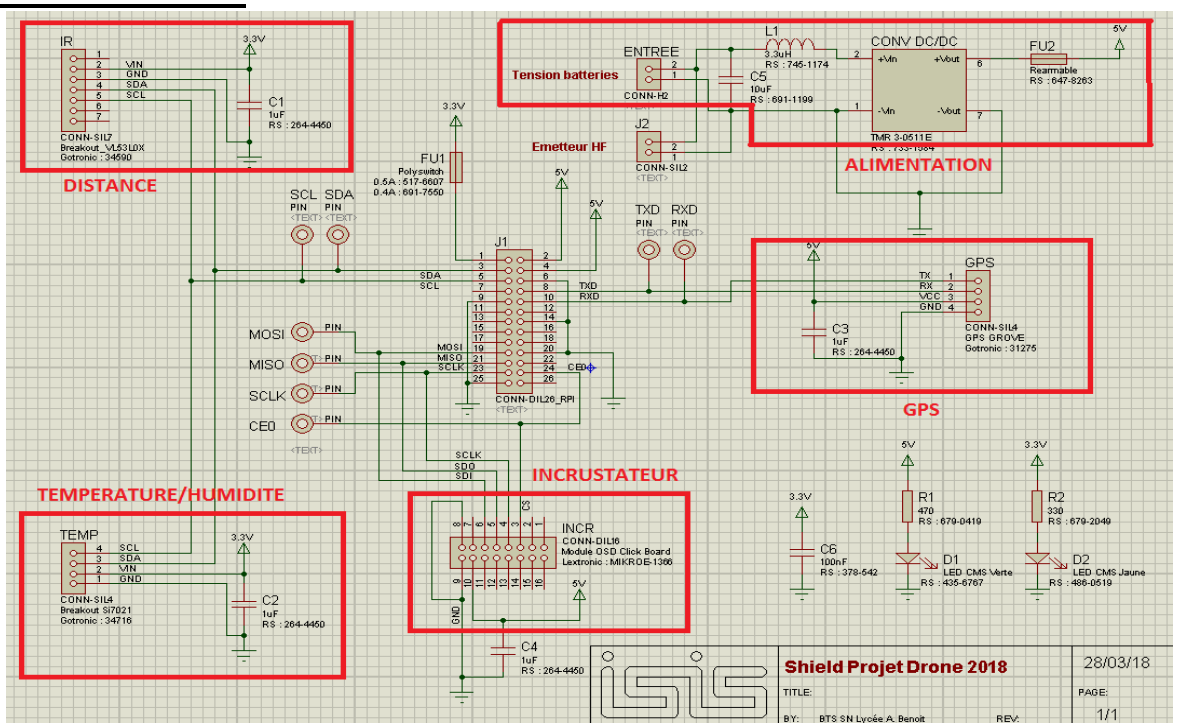
Tout à fonctionné correctement et nous avons pu faire fonctionner tous nos capteurs en même temps (voir photo), sans problème particulier a part que des fois la raspberry affiche un petit éclair qui signifie un manque d'alimentation quand elle s'allume (sûrement les condensateurs qui prennent un peu d'énergie au moment ou ont allume la Raspberry).



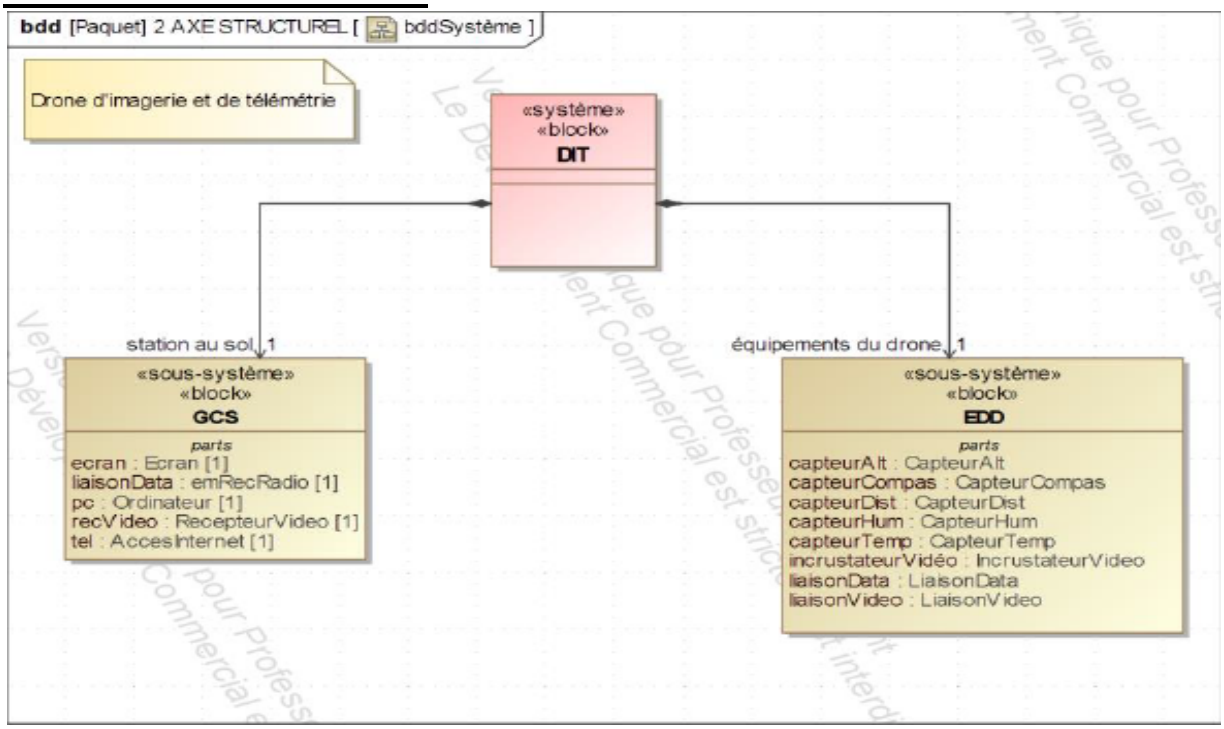
V. Routage shield et test sur carte fabriqué en cours :

Pour ce projet nous devons fabriquer un shield où nous viendront implantez dessus tous nos composants.
Dans un premier temps nous avons eu à faire un schéma structurel élaboré avec tous les membres du projet à l'aide du diagramme de blocs et le logiciel ISIS.

Schéma structurel :

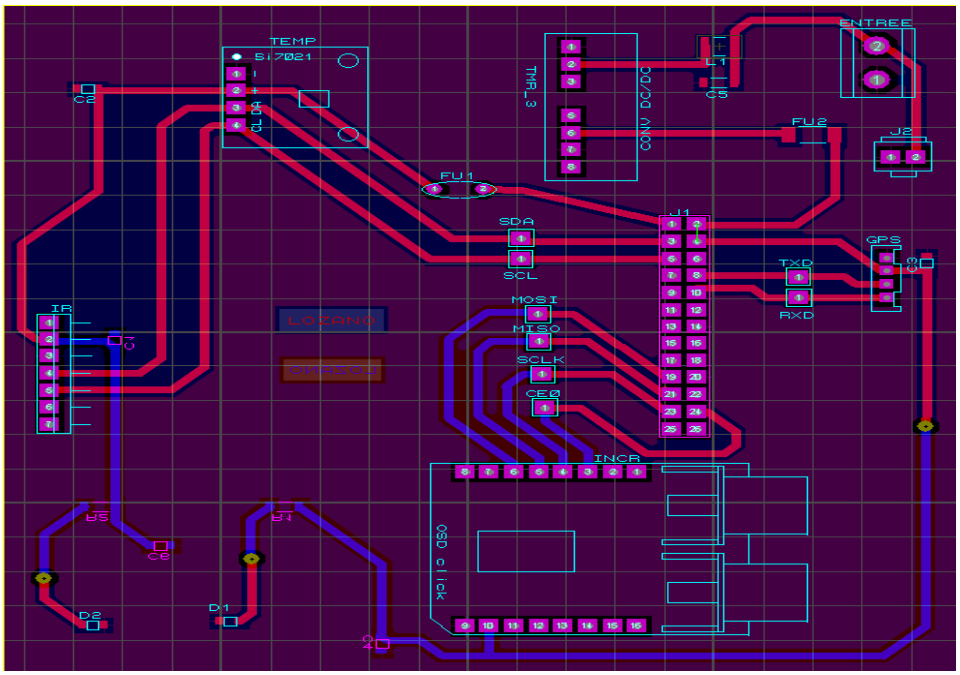


BDD du schéma structurel :

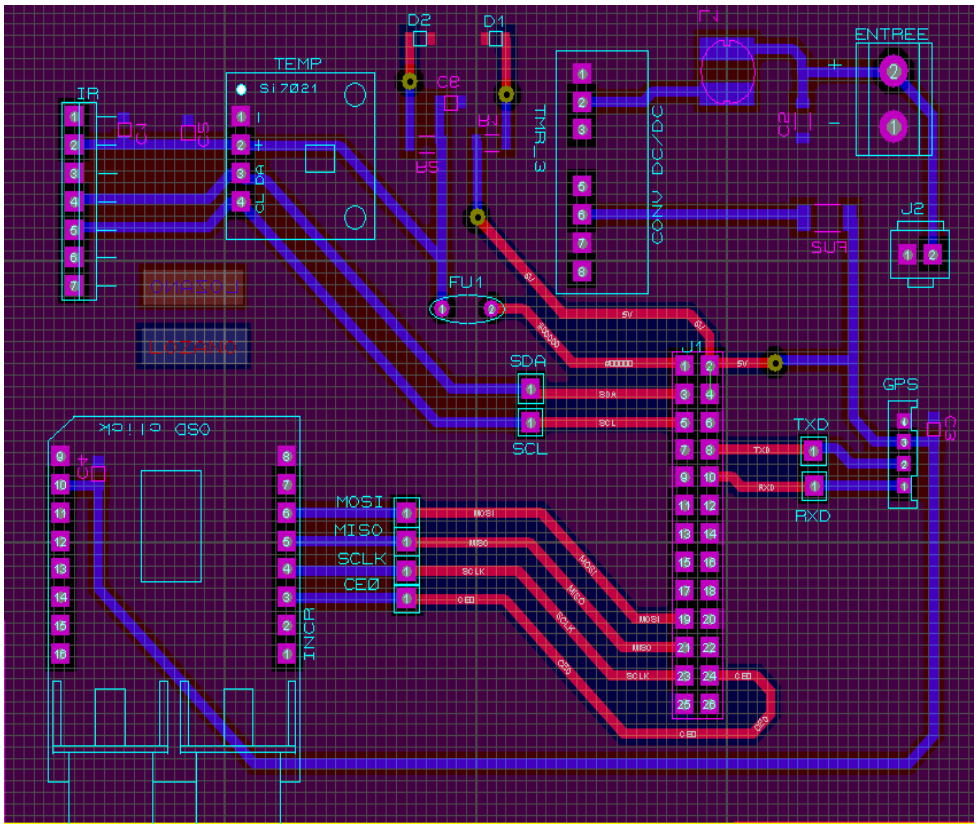


Ensuite j'ai du faire le routage à l'aide du logiciel Ares.

Première Version :

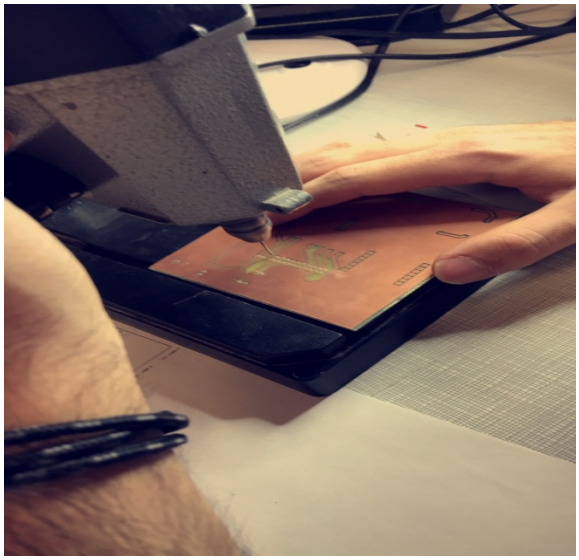


Version Finale :



Après avoir fini le routage notre professeur à imprimé l'une de nos cartes, à savoir la plus simple à concevoir pour pouvoir la tester avant d'imprimer les nôtres.

Perçage de la carte avant soudure :



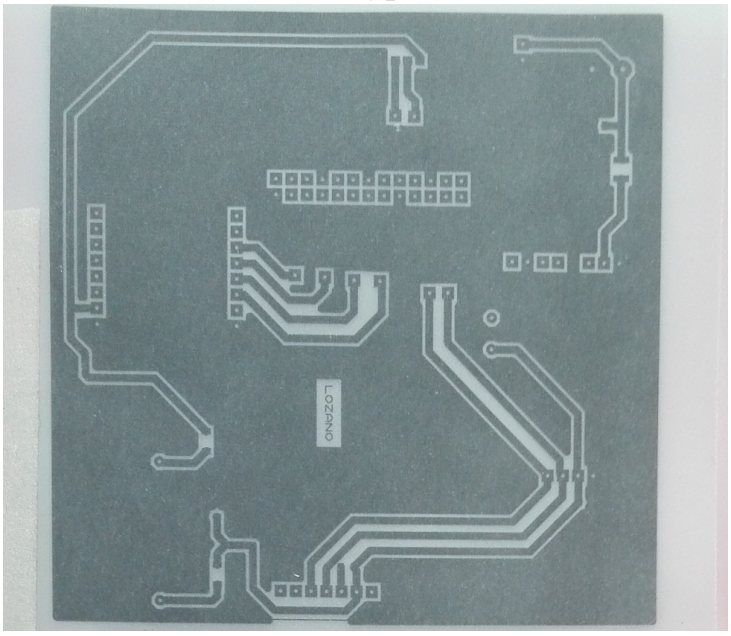
Test de la carte après soudage :

La carte n'a pas fonctionné du premier coup à cause de deux soudures qui touchaient au plan de masse mais après modifications tout fonctionnait correctement.

VI. Fabrication de ma carte :

Une fois tous les tests effectués sur la première carte imprimée en cours (Buffe) j'ai pu tester ma propre carte fabriqué par notre professeur.

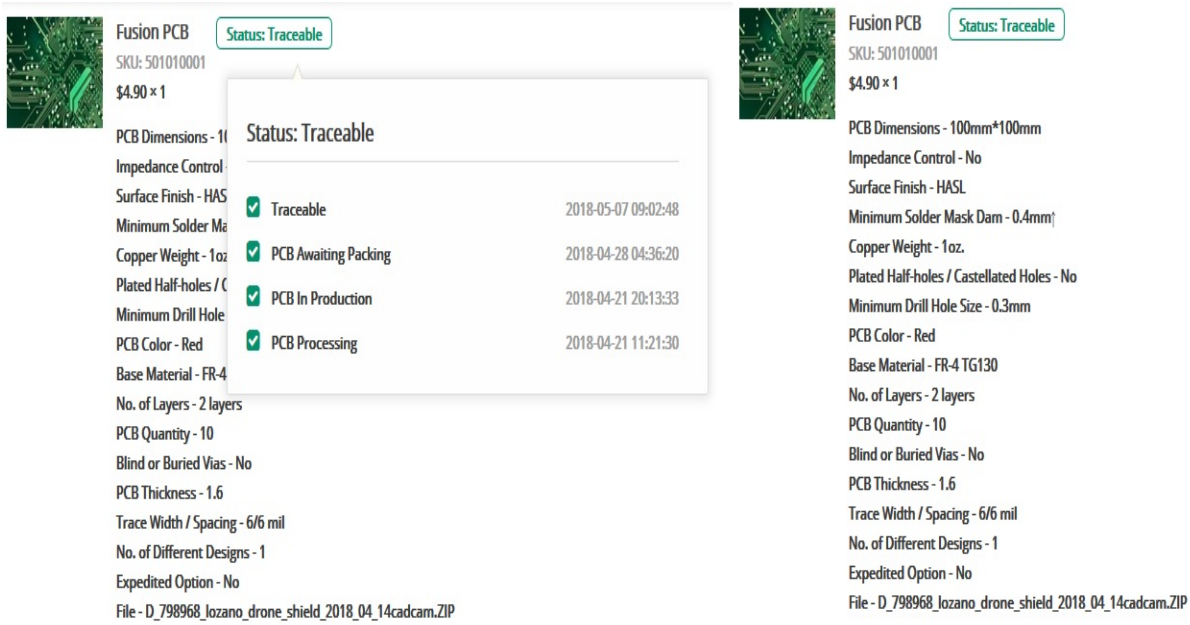
Mon Typon :



Après que mon camarade Buffe ait effectué mes soudures car je me trouvais dans l'incapacité de le faire, j'ai pu tester ma carte. Tout fonctionnait bien le seul problème c'est que j'ai dû réduire sa taille car elle prenait beaucoup de place et j'ai dû rajouter une rallonge pour les broches qui allait sur le GPIO car sinon ça touchait au niveau des ports USB sur la Raspberry.

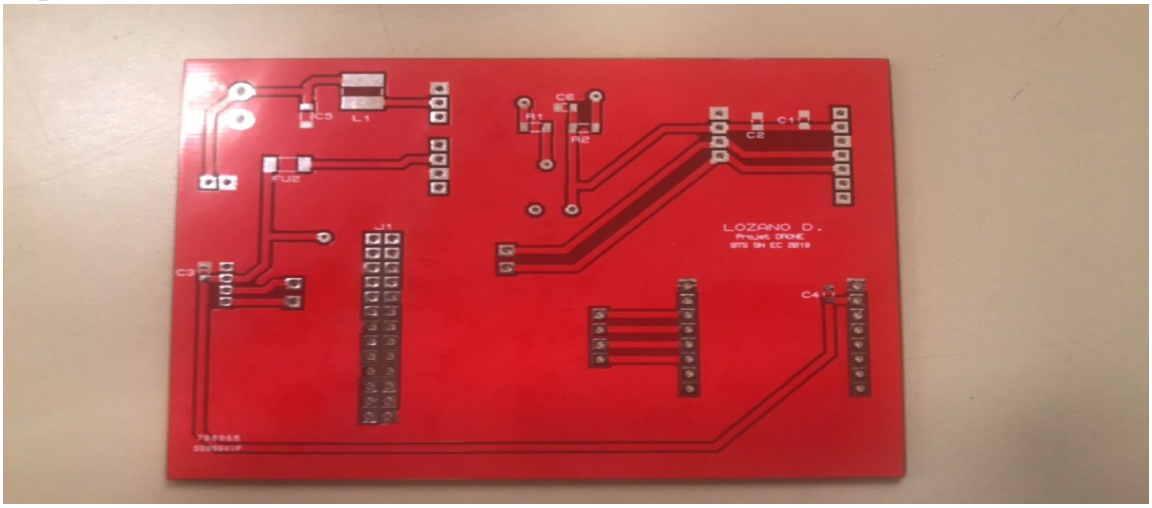
VII. Commande de la carte fabriquée par Seeed Studios et réception :

Une fois les tests terminés et après avoir modifié mon routage, j'ai pu commander ma carte et la faire fabriquer.



Entre la commande et la réception le délai à été d'environ 3 semaines, le prix pour la fabrication de cette carte à était de 4.90$ soit environ 4,10 euro pour un lot de 10 cartes.

Sur l'image au-dessus on peut voir le détail de la carte, avec notamment l'épaisseur de la carte, le nombre de couches et la couleur de la carte.



VIII. Soudage de la carte reçu et mise en service :

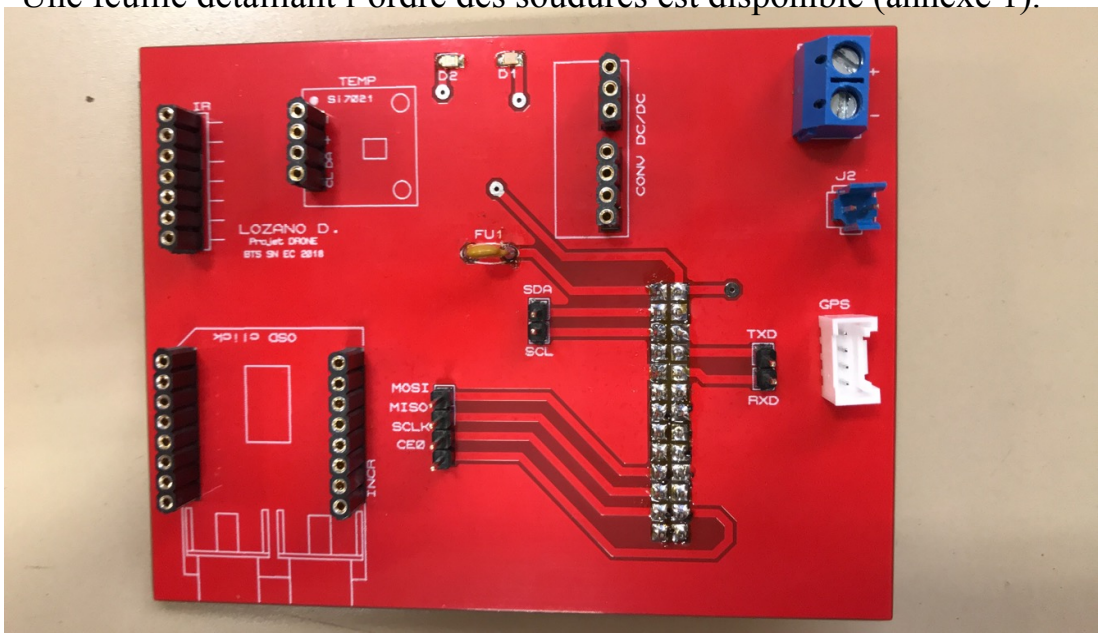
Après avoir reçu ma carte j'ai soudé tous les composants, dans un certain ordre afin de facilités les soudures.

J'ai d'abord commencé par les composants CMS de la couche top, car c'est la couche ou il ou il n'y avait pas la bobine qui est plus épaisse que les autres composants CMS, en mettant de la pâte a braser et en passant la carte au four à refusion.

Ensuite j'ai fait la même chose avec les composants CMS de la couche bottom.

Puis j'ai soudé a la main les composants traversant en allant du plus petit au plus gros.

Une feuille détaillant l'ordre des soudures est disponible (annexe 1).



Après avoir effectué les soudures et vérifier chaque soudures à l'aide d'un multimètre en position test diode pour s'assurer de la connectivité des composants, j'ai testé ma carte.

Les feuilles détaillant la mise en marche et la maintenance du système sont disponible (annexes 2 et 3).

La carte et tous les composants ont marchaient du premier coup, aucun problèmes à signaler, tous les capteurs fonctionnent bien, et en même temps.

Annexes :

1. Soudures

2. Procédure mise en marche du système, recette

3. Dépannage, maintenance

4. Diagramme de gantt

5. Nomenclature

6. Plaque en contreplaqué

1. Etapes soudures :

1- Identifier la couche bottom et top.

2- Identifier l'emplacement des composants CMS de la couche top.
(D2, D1)

3- Appliquer la pâte à braser sur la couche top à l'emplacement des CMS.

4- Disposer les composants CMS, s'assurer du bon sens des led D2 et D1 (cathode à la masse).

5- Mettre au four à refusion la carte programme 2.

6- Sur la couche bottom repérer l'emplacement des CMS, appliquer la pâte à braser.
(C1 à C6, R1 et R2, L1, FU2)

7- Disposer les composants CMS.

8- Mettre au four à refusion la carte programme 2.

9- Ensuite souder les composants traversant, des plus volumineux aux moins volumineux.

Programme 2 : Préchauffage à 90°C, 2 min 30 montée a 220°C, 1 min montée 250°C (refusion) , 1 min 50 refroidissement.

2. Première mise en service du système :

1- Brancher le convertisseur DC/DC et ensuite alimenter la carte.

2- Vérifier que la led du 3,3V (jaune) s'allume.

3- Après vérification, éteindre l'alimentation et brancher la Raspberry et alimenter la carte.

4- Vérifier que la led du 5V (verte) s'allume.

5- Après vérification, éteindre l'alimentation et brancher tous les composants et alimenter la carte.

6- Vérifier que les modules disposant d'une led s'allument (antenne HF, GPS, Incrustateur).

7- Après vérification tester les capteurs en lançant leurs codes.

8- Une fois les tests terminés pensez à éteindre la Raspberry, puis une fois qu'elle est bien mise hors tension éteindre l'alimentation.

3. Dépannage de la carte :

Si la led 5V D1 (Verte) ne s'allume pas :
-Vérifier son sens (hors tension) .
- Vérifier la piste et les soudures de la led avec un multimètre en position test diode (hors tension).
-Vérifier si il y a la sorti du convertisseur DC/DC le 5V.

Si la led 3.3V D2 (Jaune) ne S'allume pas :
-Vérifier que la raspberry est bien branchée et alimentée.
-Vérifier le sens de la led(hors tension).
-Vérifier la piste et les soudures de la led avec un multimètre en position test diode (hors tension).
-Vérifier la sortie des broches de la raspberry,

Si les modules ne sont pas alimentés :
-Vérifier que l'alimentation arrive jusqu'aux broches des connecteurs des capteurs avec un multimètre en position Voltmètre.
-Vérifier que les modules sont dans le bon sens.
-Vérifier les soudures des connecteurs pour les capteurs avec un multimètre en position test diode (hors tension).
-Vérifier la piste des capteurs avec un multimètre en position test diode (hors tension).

Si un module est alimenté mais que sont code ne fonctionne pas :
-Vérifier que le capteur est bien connecté.
-Vérifier les soudures des connecteurs sur la carte avec un multimètre en position test diode (hors tension).
-Vérifier que les pistes de la carte sont bien connectées à la sortie des broches des connecteurs avec un multimètre en position test diode (hors tension).

4. Diagramme de gantt :

Diagramme Prévisionnel :

Tâches Lozano	184 h	Jeu 11/01/18	Mer 16/05/18		LOZANO Dorian
Recherche documentation + solution de l'année précédente	20 h	Jeu 11/01/18	Ven 19/01/18		
Diagramme de blocs et de séquence	10 h	Mer 24/01/18	Jeu 25/01/18	52	
Test du capteurs sur Arduino et Raspberry	10 h	Mer 31/01/18	Jeu 01/02/18	53	
Codage	30 h	Ven 02/02/18	Mer 21/02/18	54	
Realisation de la carte alimentation	40 h	Mer 14/03/18	Jeu 29/03/18	55	
Mise en place	20 h	Jeu 29/03/18	Mer 11/04/18	56	
Routage	20 h	Mer 11/04/18	Mer 09/05/18	57	
Finalisation	20 h	Jeu 10/05/18	Mer 16/05/18	58	

Diagramme Réel :

Tâches Lozano	207 h	Jeu 11/01/18	Mar 12/06/18		LOZANO Dorian
Recherche documentation + solution de l'année précédente	25 h	Jeu 11/01/18	Mer 24/01/18		
Test du capteurs sur Arduino et Raspberry	15 h	Mer 31/01/18	Mer 07/02/18		
Codage et compréhension	14 h	Mer 07/02/18	Mer 14/02/18		
test alim avec recherche et documentation	20 h	Mer 14/02/18	Ven 23/02/18		
Developement carte sur isis	20 h	Jeu 22/03/18	Ven 30/03/18		
Nomenclature	3 h	Mer 04/04/18	Mer 04/04/18		
Routage	20 h	Jeu 05/04/18	Ven 13/04/18		
Test Carte imprimmer en cours	15 h	Jeu 12/04/18	Ven 20/04/18		
Rédaction dossier et diaporama	22 h	Ven 11/05/18	Ven 18/05/18		
Finalisation dossier	20 h	Mar 22/05/18	Ven 25/05/18		
Soudage carte reçu avec test	33 h	Mer 30/05/18	Mar 12/06/18		

5. NOMENCLATURE :

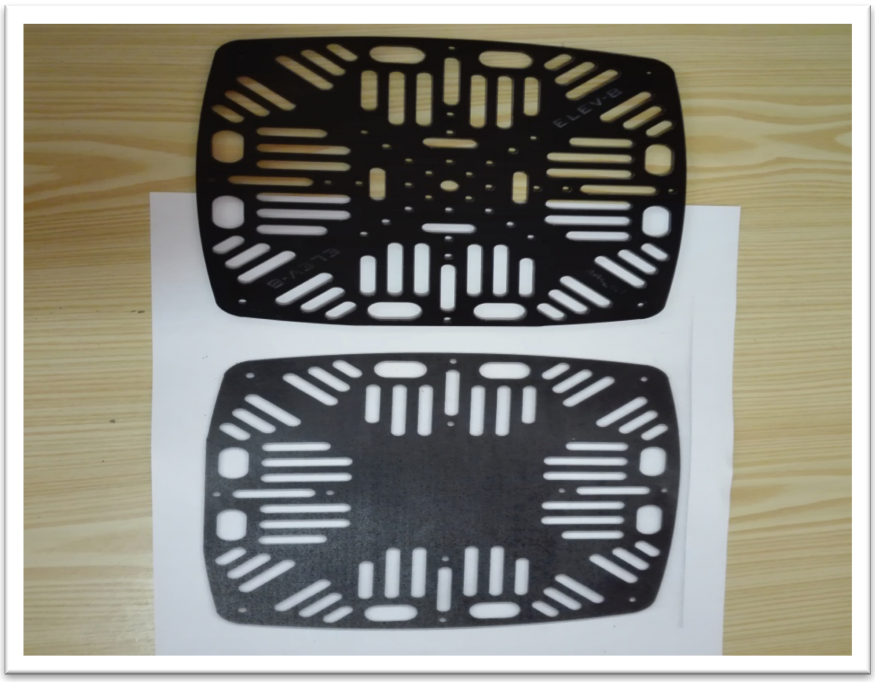
	Projet Drone 2018							Client :				
Repère	Désignation du matériel	Valeur (Référence)	Tol.±	Equivalent	Fabricant @ Fournisseur	Boîtier	Recommandation câblage	Qté	Condition	ref rapide "RS" (FA)	Prix U HT	Prix T HT
R1	Résistance CMS	470 Ohms	1 %		Radiospares	603		1		679-0419	0,019	0,019
R2	Résistance CMS	330 Ohms	1 %		Radiospares	1206		1		679-2019	0,014	0,014
C1 à C4	Condensateur CMS	1 µF	80 %		Radiospares	805		4		264-4450	0,052	0,208
C5	Condensateur CMS	10 µF	10 %		Radiospares	1206		1		691-1199	0,199	0,199
C6	Condensateur CMS	100 nF	20 %		Radiospares	805		1		378-542	0,078	0,078
D1	LED CMS Verte	HSMG-C170			Radiospares	2012		1		435-6767	0,132	0,132
D2	LED CMS Jaune	HSMY-C170			Radiospares	2012		1		486-0519	0,137	0,137
FU1	Polyswitch fusible réarmable	500mA – 0,25A			Radiospares			1		517-6607	0,333	0,333
FU2	Fusible CMS réarmable	4A – 2A			Radiospares			1		647-8263	0,417	0,417
L1	Inductance de choc	3,3µH	20 %		Radiospares			1		745-1174	2,200	2,200
PIN Mâle	2 Connecteur Mâle traversant	2,54mm			Radiospares			4		896-7620	0,054	0,216
PIN Femelle	Embase femelle 20 contacts	2,54mm			Radiospares			2		600-7748	1,214	2,428
IR	Capteur de distance				Gotronic			1		34590	11,290	11,290
TEMP	Capteur d'humidité et de température				Gotronic			1		34716	6,880	6,880
INCR	Module «osd click board »				Lextronic			1		MIKROE-1366	32,320	32,320
GPS	Module GPS Grove 113020003				Gotronic			1		31275	27,420	27,420
CONV DC/DC	Convertisseur DC/DC				Radiospares			1		733-1584	15,650	15,650
J1												0,000
J2	Embrase 2 contacts	2,54mm			Radiospares			1		423-2914	0,260	0,260
ENTREE	Bornier 2 contacts	3,5mm			Radiospares			1		710-0444	0,520	0,520
Port Groove	Embrase CI 4 contact	2mm			Radiospares			1		512-8608	0,401	0,401
										TOTAL HT FEUILLE		101,122
										TOTAL HT GLOBAL		101,122
										COEFFICIENT		0,000

	Nom	Date	Folio
Crée :	TISON Sylvain	05/04/2018	1/1
Mis à jour :	TISON Sylvain	11/05/2018	

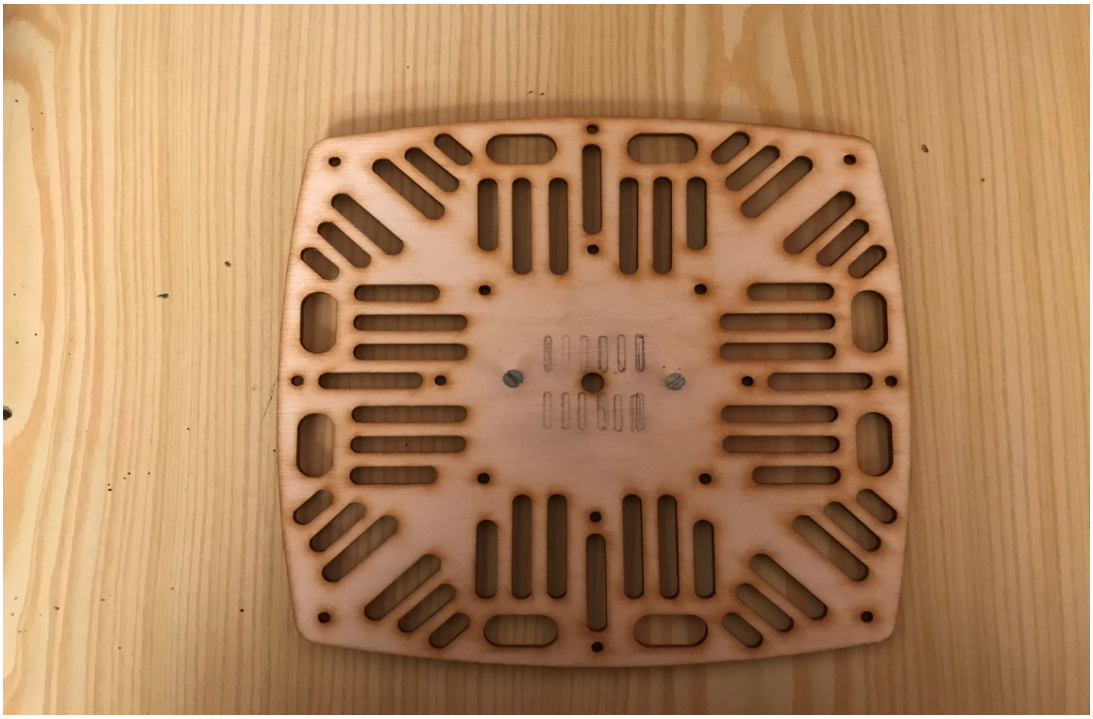
6. Plaque en bois fabriquée par les ITEC :

Pour mettre notre Raspberry avec notre carte d'extension, nous avons choisi de faire appel à la section ITEC pour fabriquer une plaque en contre-plaqué.

Scan de la plaque pour leurs logiciels d'impression :



Résultat :



EC 2 : Incrustateur Vidéo
Buffe Jordan

I - Présentation :

J'appartiens à l'équipe 1 du projet drone, elle est composée de 2 EC et 2 IR.

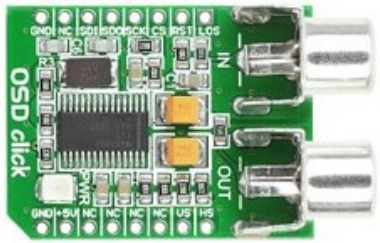
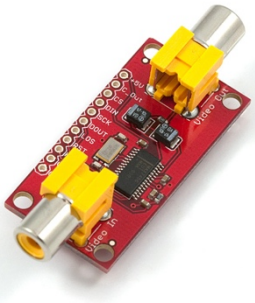
Mon rôle au sein du projet est de faire afficher sur un écran et à l'aide d'un incrustateur les différentes informations relevées sur le drone. Ces informations seront la température, les coordonnées GPS ainsi que la distance séparant le drone d'un obstacle.

Ces parties seront liées aux travaux de Dorian Lozano (Température), Sylvain Tison (GPS) et Nicolas Bottigliero (Distance).

Les cartes qui permettent l'incrustation vidéo sont composées d'un microcontrôleur MAX7456 en effet celui-ci est optimisé pour l'incrustation. La carte utilisée les années précédentes n'est plus fabriquée il faudra alors que j'en trouve une répondant aux exigences. Je ferai alors un comparatif qui me permettra de trouver la plus apte à correspondre. Il faudra au préalable que je prenne connaissance du travail de l'année précédente avec l'ancienne carte.

Étudiant	*Liste des fonctions assurées par l'étudiant :*	*Installation* : - Rpi : bus SPI et librairie C/C++ BCM2835. *Mise en œuvre* : - Utiliser la documentation de la version précédente du projet, pour mettre en œuvre l'incrustateur utilisé en 2017*(Sparkfun)*. - Valider par prototypage rapide l'incrustateur proposé pour la version 2018 *(OSD click board MikroElectronika)*. *Réalisation* : - Participer avec tous les autres étudiants EC à la conception d'un schéma commun intégrant tous les composants de l'EDD. Outre l'incrustateur, la partie émission vidéo sera liée à ce contrat. - Concevoir une carte électronique personnelle compatible avec tous les contrats EC. *Configuration* : - Mettre à jour la table des caractères de l'incrustateur pour intégrer les caractères nécessaires aux unités des mesures à incruster. *Documentation* : - Bibliothèque C/C++ d'accès à l'incrustateur. - Documents de fabrication de la carte. Ces documents devront permettre une fabrication industrielle du circuit imprimé.
EC2 IR	EDD : UC Incruster Etudier la solution 2017. Valider une nouvelle solution. Concevoir/Réaliser/Tester une nouvelle carte d'extension intégrant tous les breakout des capteurs et autres. Développer une application permettant de vérifier le bon fonctionnement de l'incrustateur.	

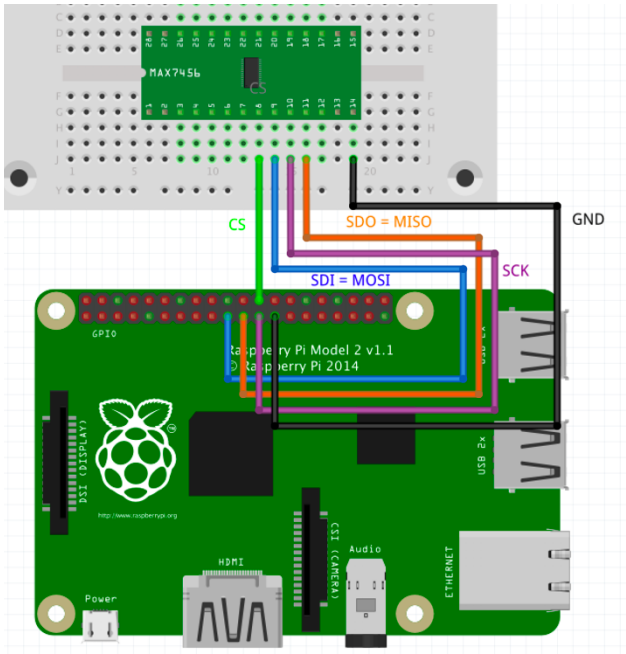
II - Comparatif :

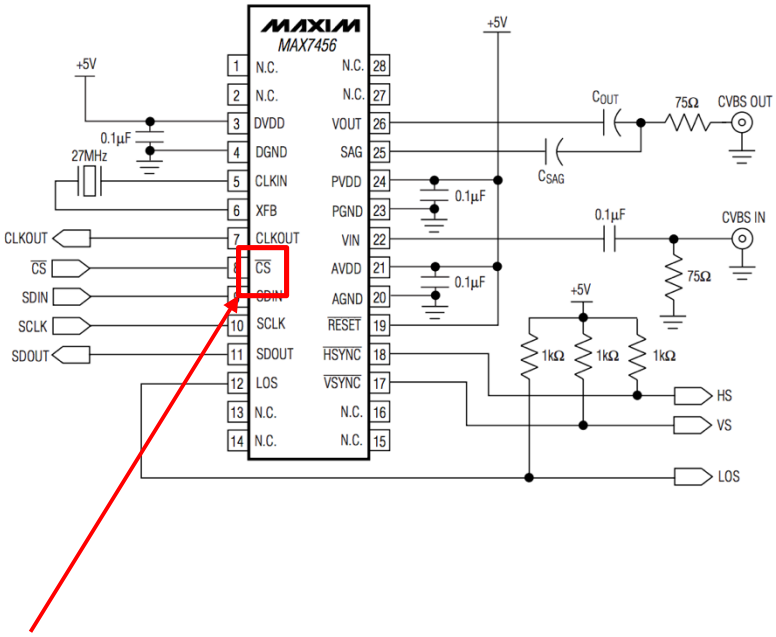
OSD Click Board	SparkFun On Screen Display Breakout	MAX7456 OSD Breakout	OM9024
			
Petit Simple 38,78€	Petit Simple N'existe plus Modèle des années précédentes	16,90€	Grande Pas adapter pour le drone Optimiser pour surveillance vidéo 72,92€
MAX7456	MAX7456	MAX7456	OM9024 Affichage couleur
Interface : SPI Alimentation : 5V	Interface : SPI Alimentation : 5V Modèle plus fabriquer donc plus envisager.	Interface : SPI Alimentation : 5V Pas d'entrer vidéo seulement grove	Interface : SPI, RS232, TWI Alimentation : 7~9V

On choisira le module OSD click board, car celui-ci est simple de mise en place et revient le moins cher.

III - Schéma de câblage rapide du MAX7456 avec une Raspberry :

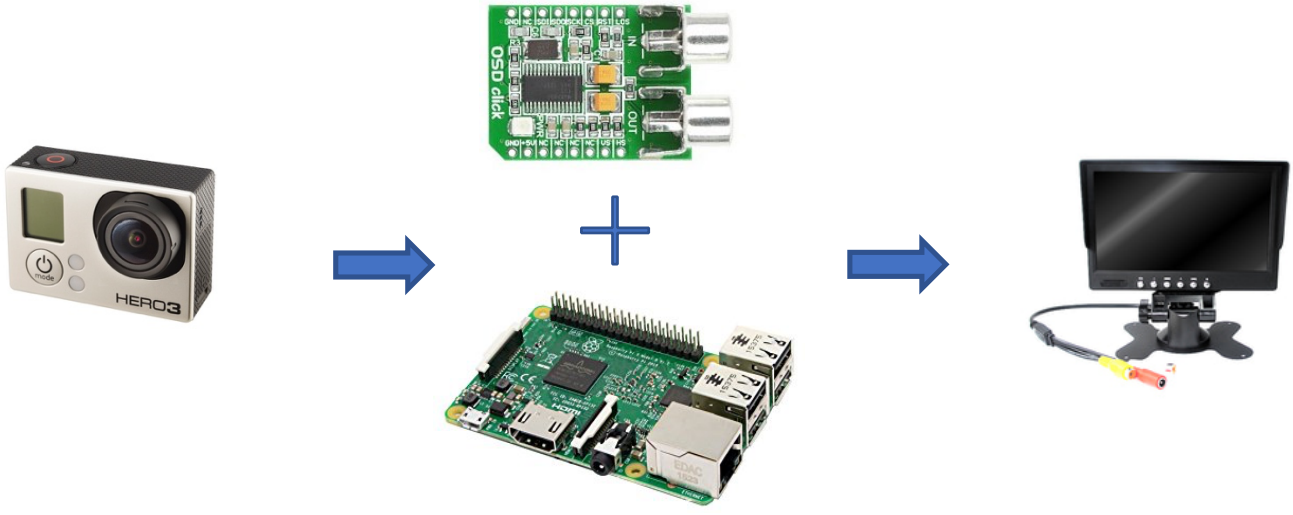
Après une prise de connaissance du site *f5mna.free.fr* et de la documentation technique du MAX7456 on peut en déduire le schéma suivant.





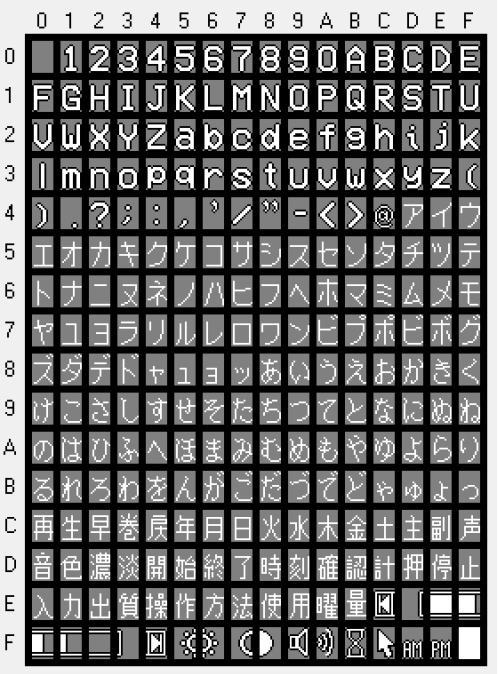
Le MAX7456 marche sur un bus SPI donc la broche CS (chip selection) doit être à l'état bas pour être active. Il permet de choisir le composant avec lequel on veut communiquer, ici la question ne se pose pas puisque nous n'avons qu'un seul composant.

Voici la solution envisagée pour simuler le vol aérien en attente de connexion sans fil.



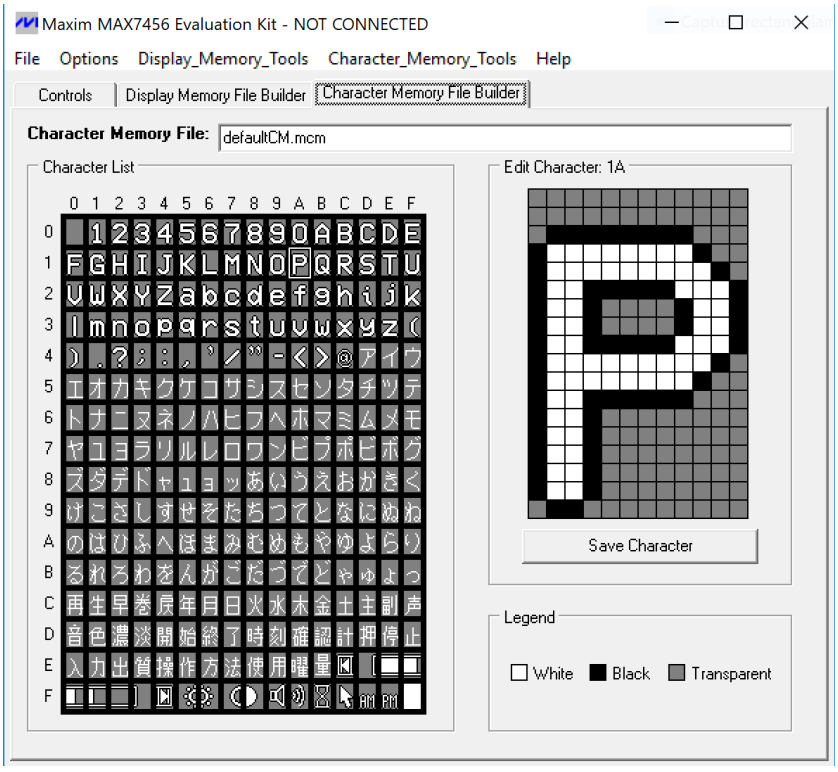
IV - Incrustation de caractères particuliers :

Le MAX7456 est régis par une table de caractères qu'il lui est propre.

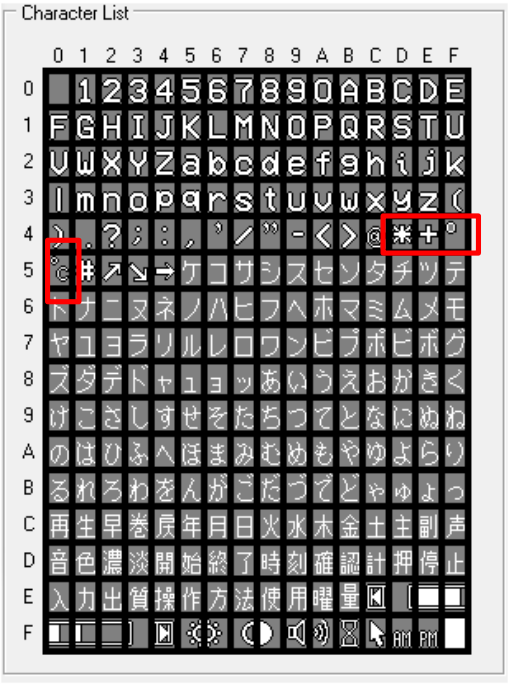


Ici la table par défaut, cependant pour correspondre aux exigences de notre projet il manque certains caractères tels que « °C » pour afficher la température sur l'écran.

Pour se faire je me suis servi du logiciel fourni par le fabricant. Permettant la modification et la synchronisation de la nouvelle table.



Après l'avoir créé j'ai pu l'insérer sur le MAX7456. J'ai alors utilisé le code fourni pour la transférer, à cause de la complexité j'ai été contraint de transférer la table par l'intermédiaire d'une arduino. En effet l'envoie par la raspberry me demande la réécriture complète du code et je ne dispose pas de ces capacités. Cependant j'ai pu modifier le code originel sur l'arduino pour pouvoir utiliser les nouveaux caractères.

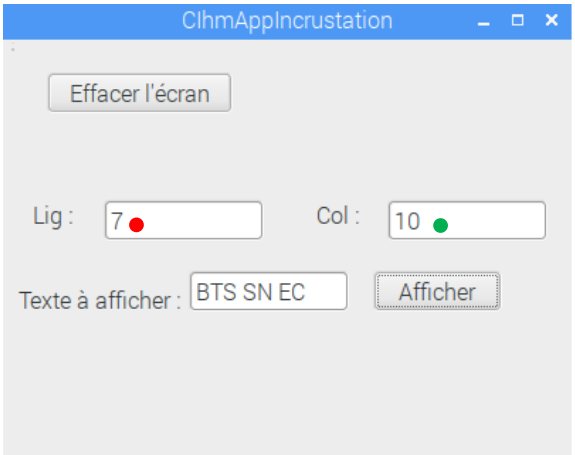



```
  else if (character == 43)
    lookup_char = 0x4e; // +
else if (character == 42)
    lookup_char = 0x4d ; // *
else if (character == 35)
    lookup_char = 0x51 ; // #
else if (character == 176)
    lookup_char = 0x4f ; // °
else if (character == 94)//^
    lookup_char = 0x50 ; // °C
```

En effet certains caractères ne sont pas sur le clavier, alors pour utiliser le « °C » il faudra envoyer le caractère 94 soit « ^ ».

Une fois transférer sur le MAX7456 la table est garder en mémoire, on peut alors passer sur la raspberry et mettre le code permettant l'incrustation vidéo.

Une application a été créé par Mr Antoine, me permettant d'afficher du contenu sur l'écran. L'objectif est de modifié cette application afin d'y incruster les valeurs de la température, humidité etc…
Il faut rentrer les coordonnées pour savoir ou es ce qu'on veut afficher sur l'écran.



V - Commande « effaceEcran » :

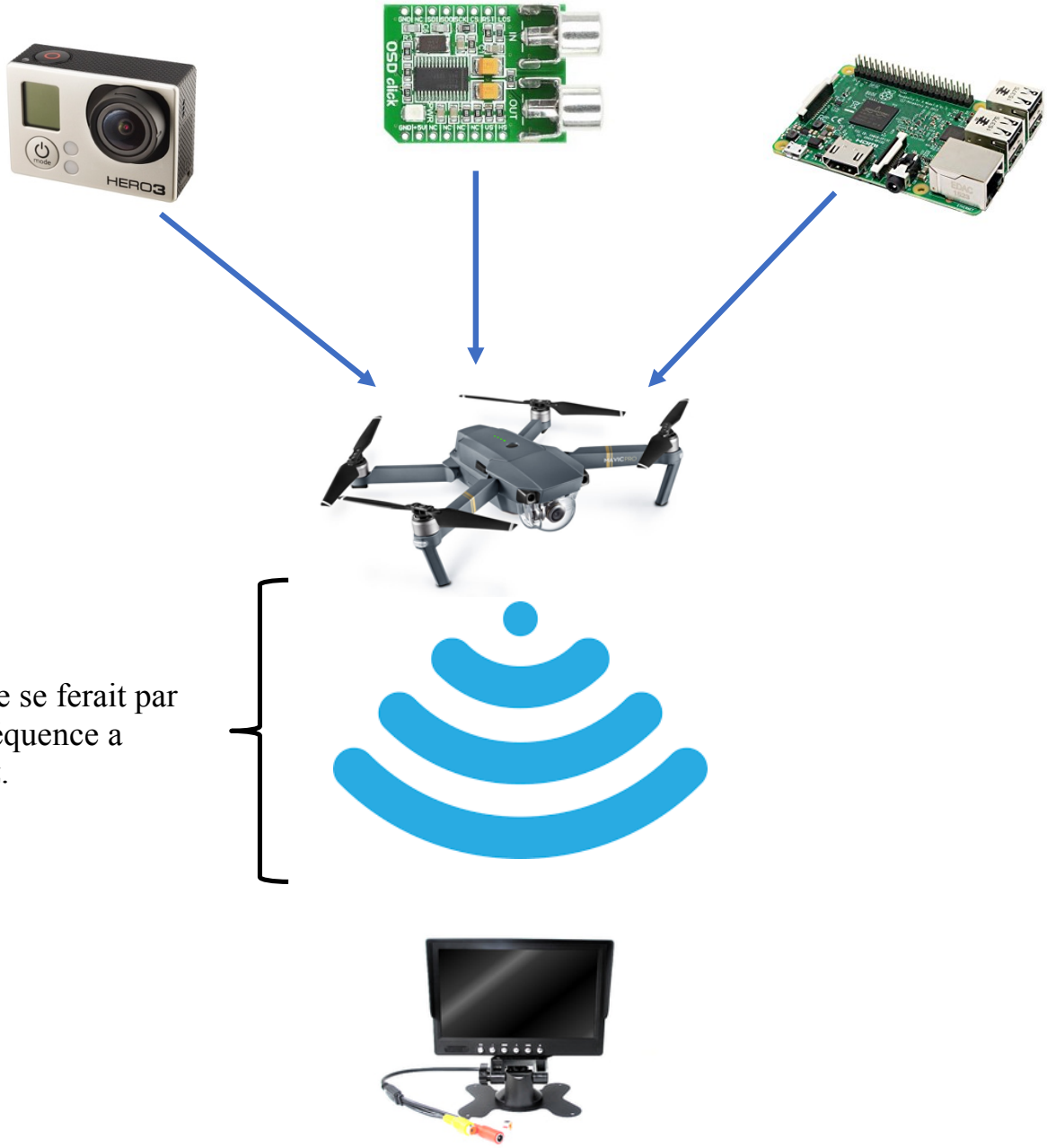
La commande « effaceEcran » est présente sur l'application, est comme son nom l'indique elle permet d'effacer le contenu de l'écran. Cet commande est très utile car lors de l'envoi successifs de texte, l'écran n'efface pas le contenu. En effet une fois le texte envoyer il est rafraichi en permanence sur l'écran, la commande « effaceEcran » à donc une grande importance.


```
int CDeviceSpiMax7456::effaceEcran()
{
    int res;
    unsigned char reg=0;
    unsigned char com[2];
    qDebug() << "CDeviceSpiMax7456::effaceEcran ";
    com[0]=DMM|READ;
    res=mSpi->ecrireNOctets(com,1);  // demande de lecture de DMM
    if (res != 1) qDebug() << "CDeviceSpiMax7456:effaceEcran:2: Pb Ecriture";
    qDebug() << "CDeviceSpiMax7456::effaceEcran : Milieu";
    usleep(1000);
    res=mSpi->lireNOctets(&reg, 1);   // lecture de DMM
    if (res != 1) qDebug() << "CDeviceSpiMax7456:effaceEcran:1 Pb Lecture";
    com[0] = DMM;
    com[1] = reg | CLEAR_DISPLAY;
    usleep(1000);
    res=mSpi->ecrireNOctets(com,2);
    if (res != 2) qDebug() << "CDeviceSpiMax7456:effaceEcran:2: Pb Ecriture";
    usleep(1000);  // temps d'effacement typiquement 20us
    qDebug() << "CDeviceSpiMax7456::effaceEcran : Sortie";
    return 1;
}
```

```
void CIhmAppIncrustation::on_pbEffacer_clicked()
{
    m_max->effaceEcran();
}
```

Ci-dessus, la partie du code permettant d'effacer l'écran. Ainsi que la partie du code appelant la méthode « effaceEcran ».
D'après la documentation technique, l'effacement de l'écran consiste à positionner à la valeur 1 le bit b2 du registre DMM du composant d'incrustation.

VI - Principe de transmission avec la station au sol :



L'envoie se ferait par radio fréquence a 5,8MHz.

En théorie on devrait utiliser le récepteur pour récupérer le signal vidéo, cependant il semblerait que l'écran dispose d'une antenne intégrer. On pourrait alors se dispenser du récepteur.

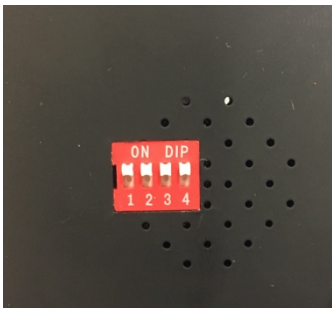
VII - Communication sans fil :

Comme l'écran intègre une antenne on peut se dispenser d'un récepteur.
Il peut recevoir des données à partir des fréquences suivantes :

CH1 : 5705 MHz
CH2 : 5685MHz
CH3 : 5665 MHz
CH4 : 5645 MHz
CH5 : 5885 MHz
CH6 : 5905 MHz
CH7 : 5925 MHz
CH8 : 5945 MHz

FR \ CH		CH							
		Ch1	CH2	CH3	CH4	CH5	CH6	CH7	CH8
FR	FR1(F)	5740	5760	5780	5800	5820	5840	5860	5880
	FR2(E)	5705	5685	5665	5645	5885	5905	5925	5945
	FR3(A)	5865	5845	5825	5805	5785	5765	5745	5725
	FR4(R)	5658	5695	5732	5769	5806	5843	5880	5917
	FR5(B)	5733	5752	5771	5790	5809	5828	5847	5866

Nous avons à droite les fréquences disponibles sur l'émetteur. La ligne FR2 correspond parfaitement aux fréquences de l'écran.



Le réglage sur l'écran se fait par l'ajustement des bagues, qui sont codées en binaires. Ici on se placera sur la station 1 soit 5,705GHz.

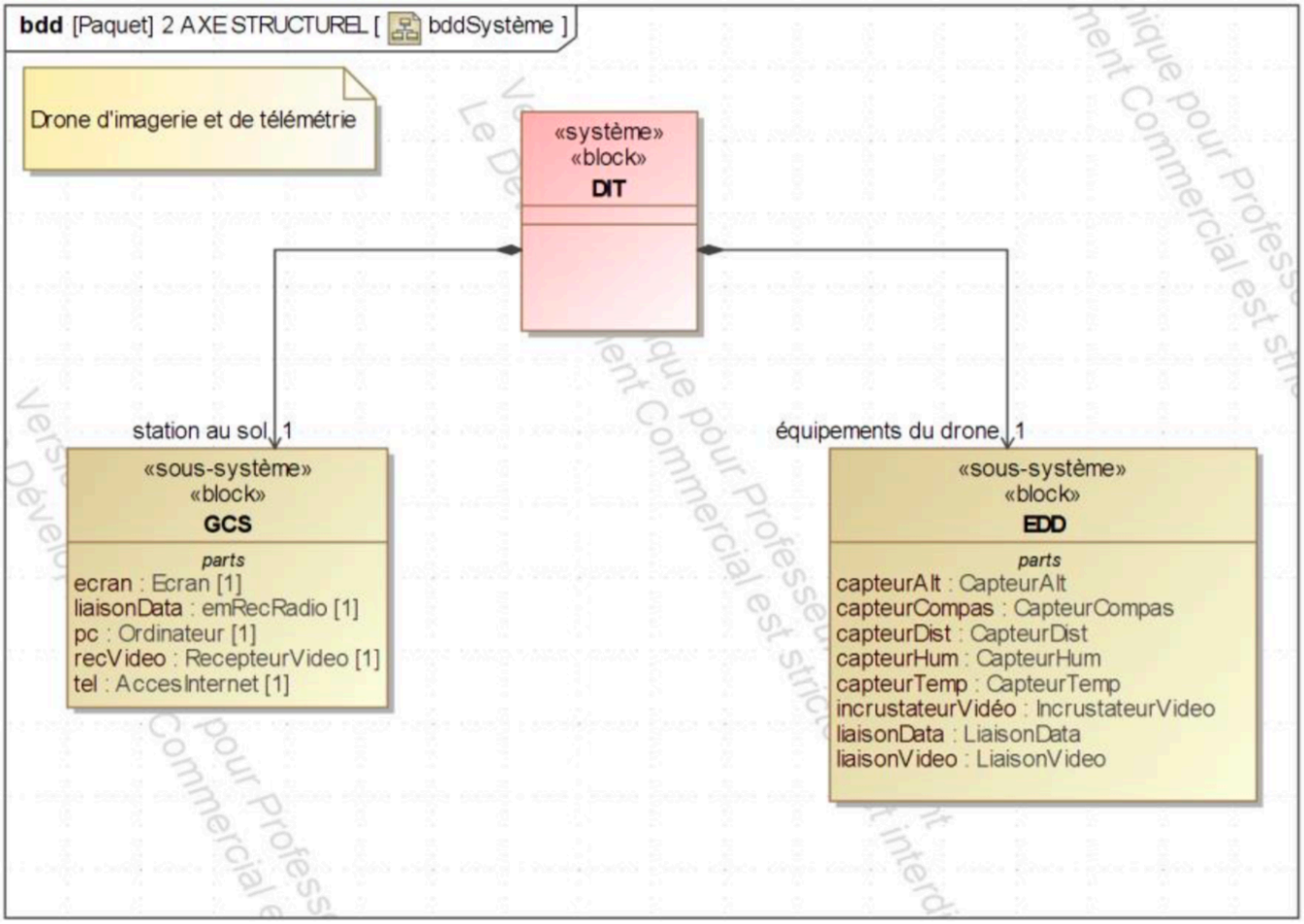
Le changement de station d'émission se fera par l'appui successifs du bouton. Une LED s'allumera pour spécifier la fréquence d'émission.

Une fois les fréquences réglées l'émetteur émet en permanence, aucun ajustement au niveau du code n'est nécessaire.

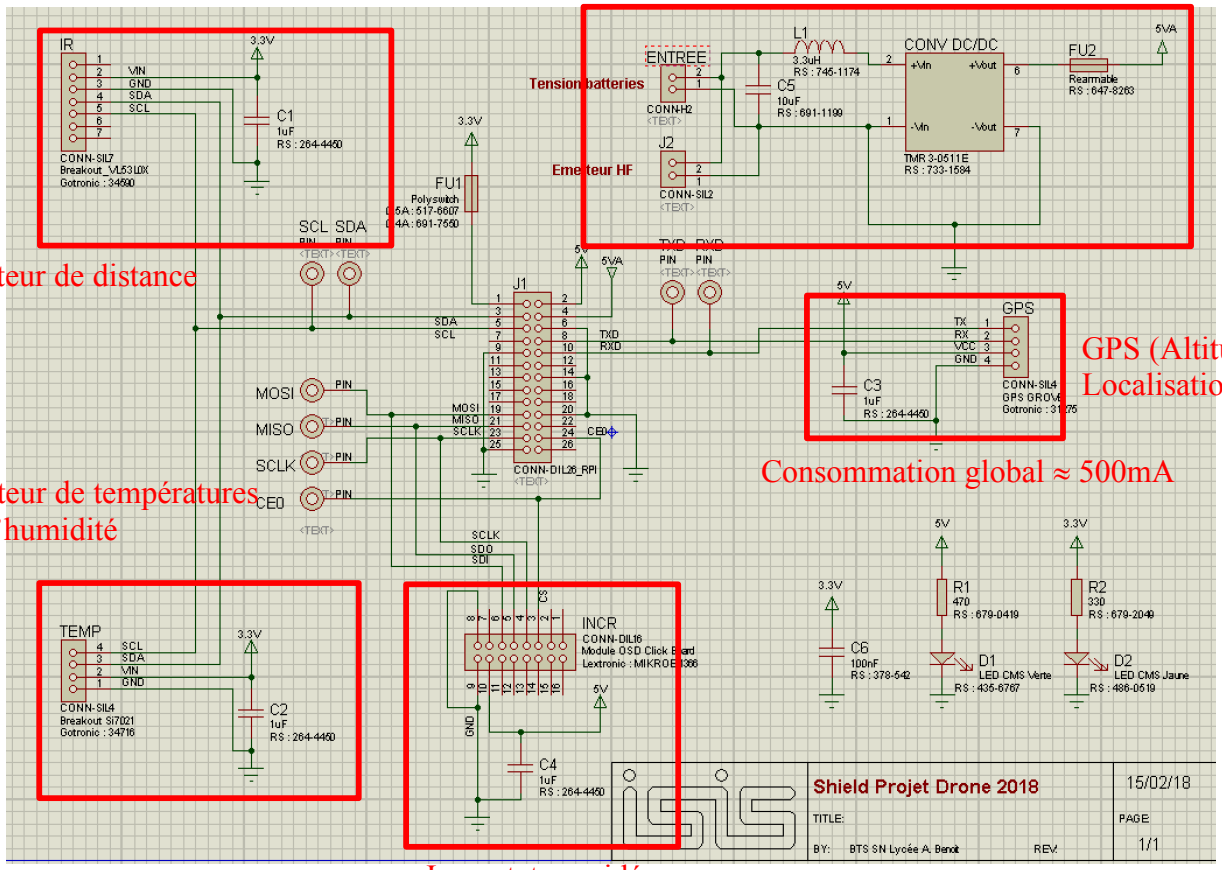
VIII - Schéma de la carte alimentation :

Le drone embarquera deux batteries, fournissant 7,2V, cela permettra d'alimenter toutes les différentes parties du drone. Nous nous intéresserons à l'alimentation de la raspberry ainsi que des différents capteurs, les moteurs ainsi que la partie du pilotage du drone ne nous concernent pas.

Après discussion avec tous les membres EC, nous avons décidé de concevoir une carte avec tous nos capteurs et qui embarquera le module permettant de fournir une tension conforme à nos capteurs. De plus Dorian Lozano, qui s'occupe de définir les composants nécessaires à l'alimentation de la carte, à trouvé que l'on pouvait alimenter la raspberry à partir du GPIO.



Module de conversion de tension 7V vers 5V



Capteur de distance

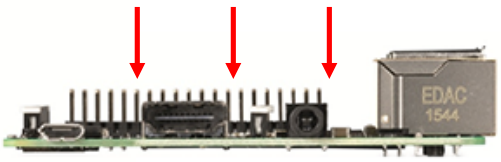
Capteur de températures et d'humidité

GPS (Altitude / Localisation)

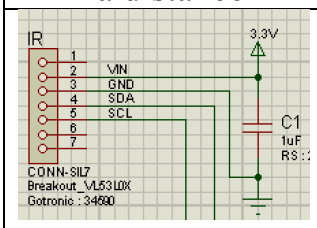
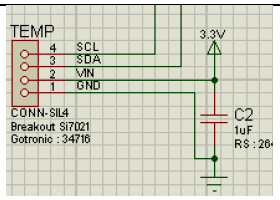
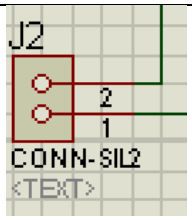
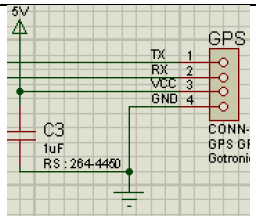
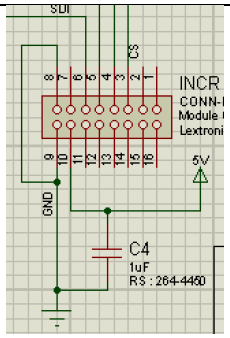
Consommation global ≈ 500mA

Incrustateur vidéo

Apres conception notre carte viendra se connecter sur la raspberry.



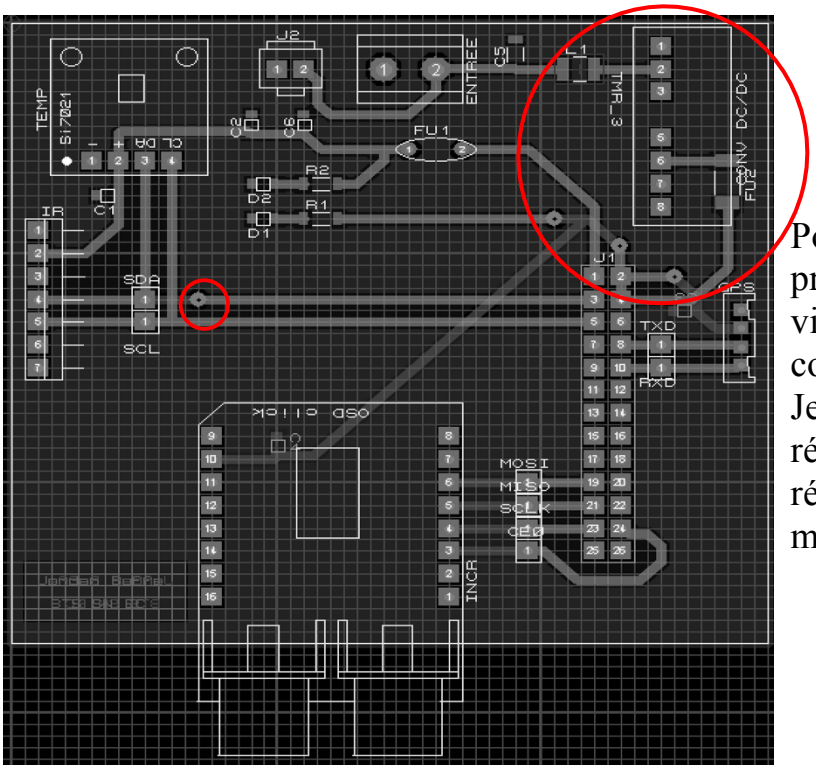
Grace au GPIO sur la raspberry nous pouvons communiquer avec tous nos différents composants. En effet il dispose de plusieurs types de BUS de communications : SPI, I2C, UART. De plus le GPIO fournis de type d'alimentation le 3,3V ainsi que le 5V.

Capteur Infra-Rouge – Détecter la distance	Capteur de Température	Broche d'alimentation de l'antenne	GPS	Incrustateur Vidéo
				
Tension d'alimentation : 3,3V Connecter à la broche 1 du GPIO.	Tension d'alimentation : 3,3V Connecter à la broche 1 du GPIO.	Tension d'alimentation : 7V-20V Grace au convertisseur DC/DC qui restitue du 7,2V on pourra alimenter l'antenne.	Tension d'alimentation : 5V Connecter à la broche 2 du GPIO.	Tension d'alimentation : 5V Connecter à la broche 2 du GPIO.
Communication : I2C	Communication : I2C		Communication : UART	Communication : SPI

IX - Routage :

1er Routage :

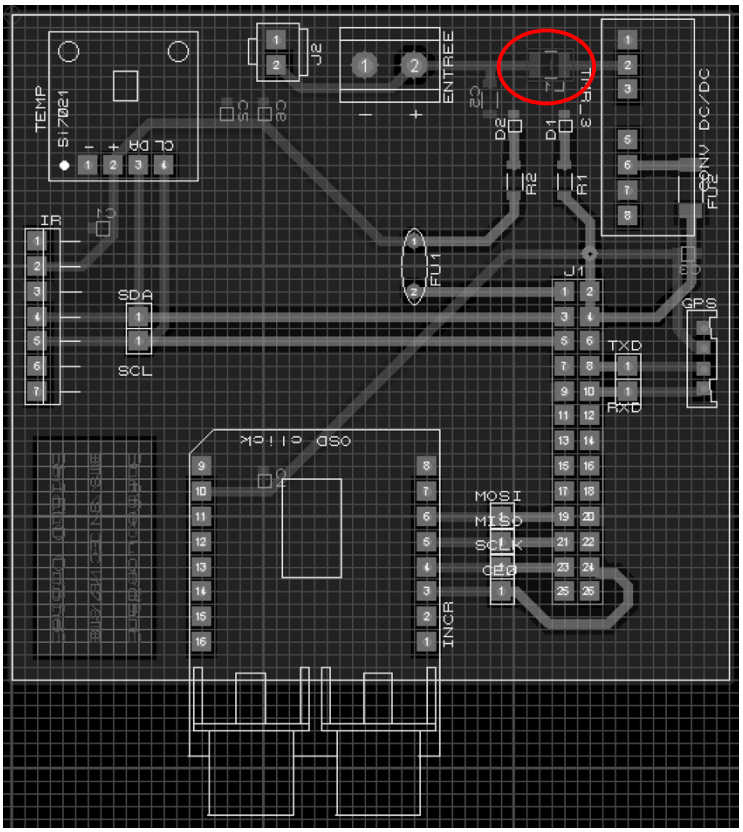
Avec le shéma fait sur ISIS j'ai pu créer le schéma de routage suivant :



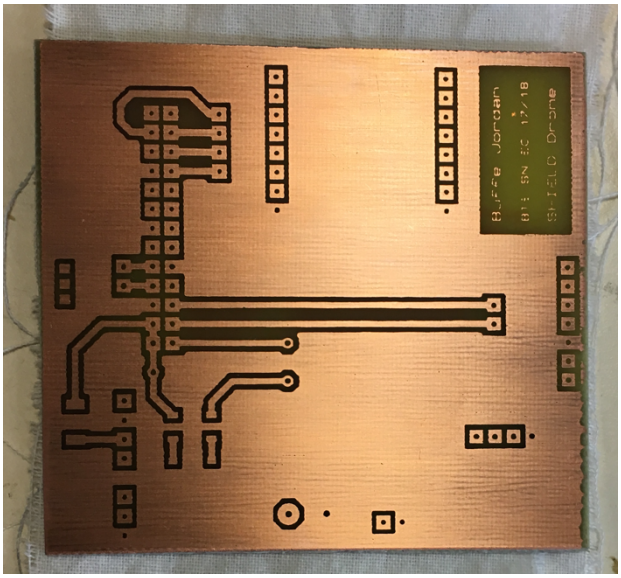
Pour mon premier routage mon professeur a estimé qu'il y avait trop de vias (traversées) pour lier les composants de la couche top a bottom. Je l'ai alors repris pour faciliter la réalisation au lycée. Car on ne peut pas réaliser des cartes avec des trous métalliser au lycée.

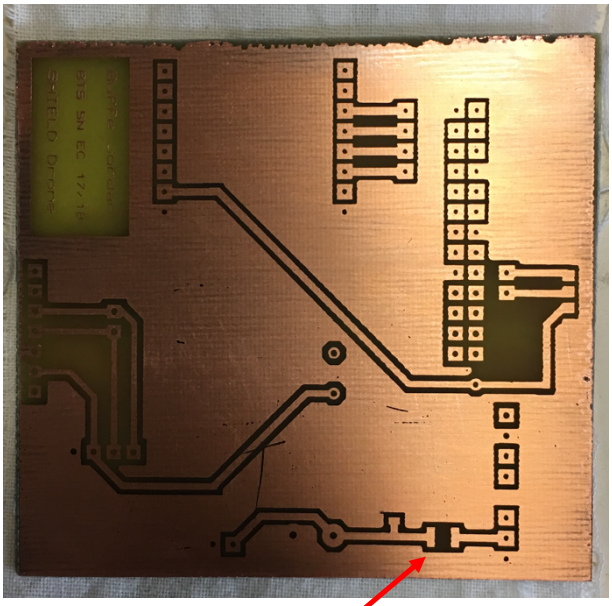
2eme Routage :

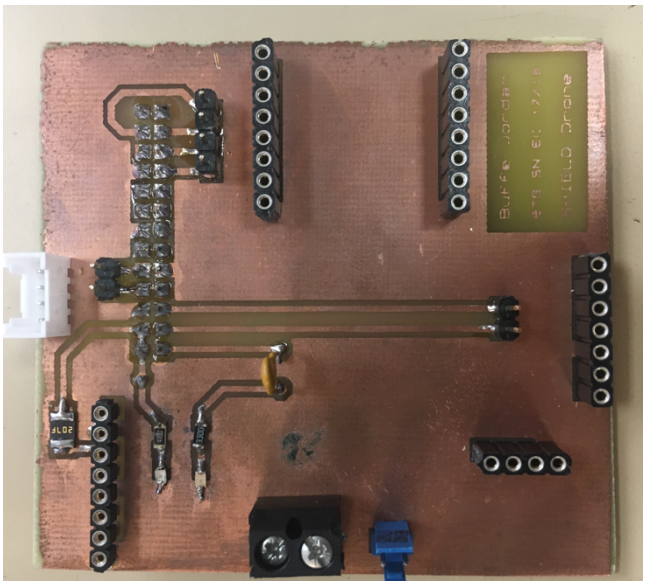
J'ai apportés un certain nombre de correctifs tels que la réorganisation des composants afin d'enlever les trous métalliser, certains composants ont été changés de coté comme la bobine L1 qui ne pouvait être reliée que par la couche bottom.

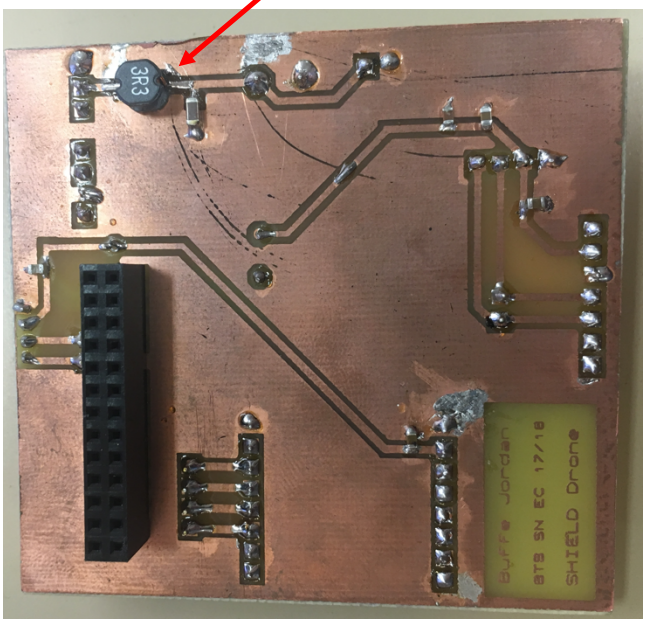


Pour tester si le schéma structurel et le routage ne comportaient pas d'erreur nous avons fait une première réalisation de ma carte car elle semblait être la plus petite à réaliser, de plus elle avait le moins de trous métallisés.







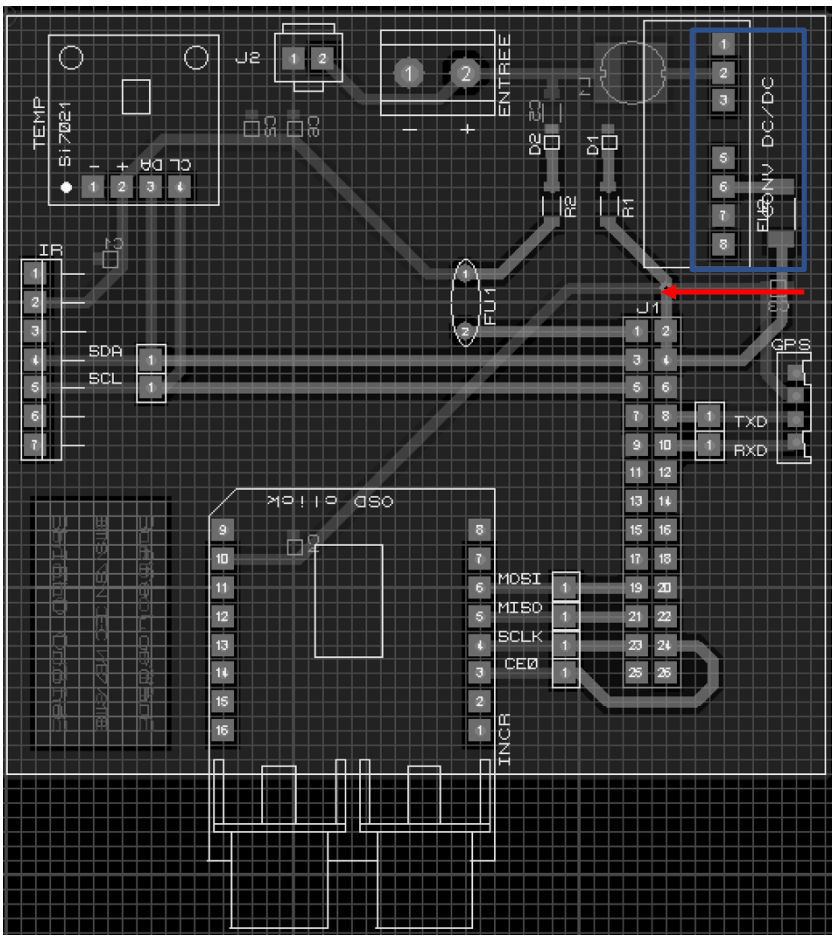


Après quelques problèmes de masse qui ont était résolus et une fois les composants soudés nous nous sommes rendu compte qu'il y avait des défauts de réalisation, les empreintes ne correspondaient pas pour la Bobine L1 ainsi que pour le convertisseur DC/DC.

3eme Routage :

Suite à ces erreurs j'ai fait un troisième routage pour fabriquer la carte dans sa version définitive sur le site www.seeedstudio.com.

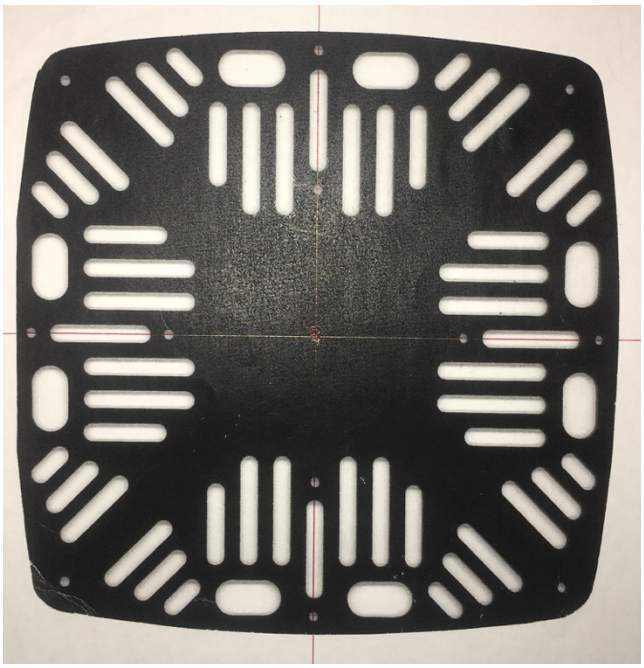
Ici on peut voir le routage final avec les corrections sur la bobine et sur le convertisseur, celui-ci était mal positionné.

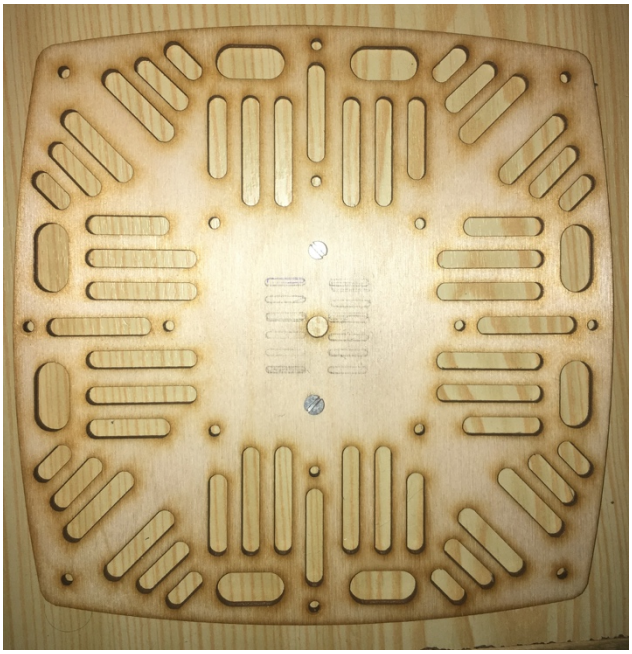


Les pastilles de la bobine ont été corrigées, ainsi que le convertisseur DC/DC sa forme a été remodelée.

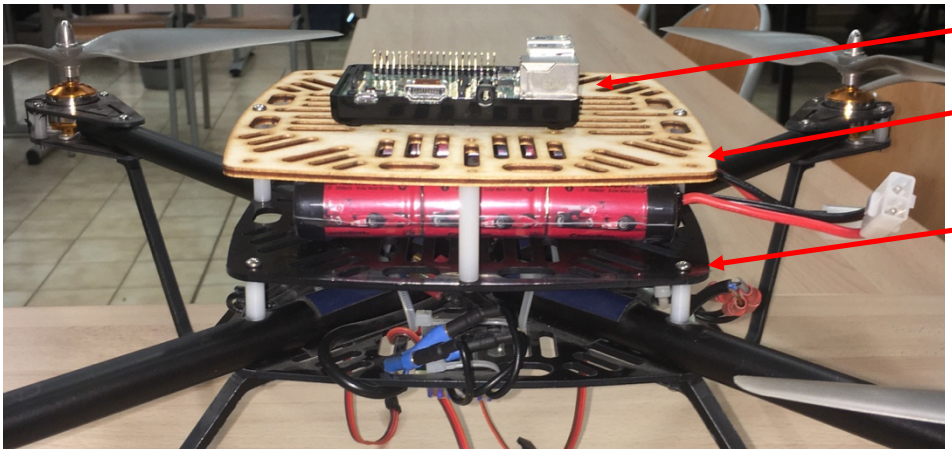
X - Plaque supplémentaire :

L'alimentation de la raspberry sera assuré par des batteries, alors, avec une concertation avec les EC, nous nous sommes rendu compte qu'il fallait de l'espace supplémentaire sur le drone. Nous avons eu l'idée de rajouter un étage sur le drone. Pour se faire nous avons fait appelle à la section STI2D ITEC pour la concevoir.



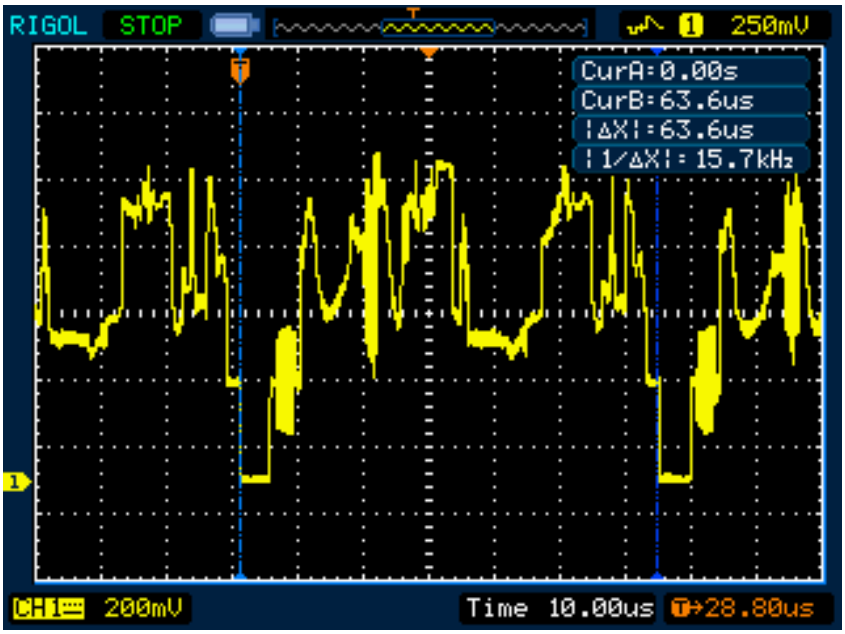


Nous leur avons transmis une image aux grandeur réels avec laquelle ils ont pu concevoir, avec une découpe laser, la plaque que l'on peut voir à droite.



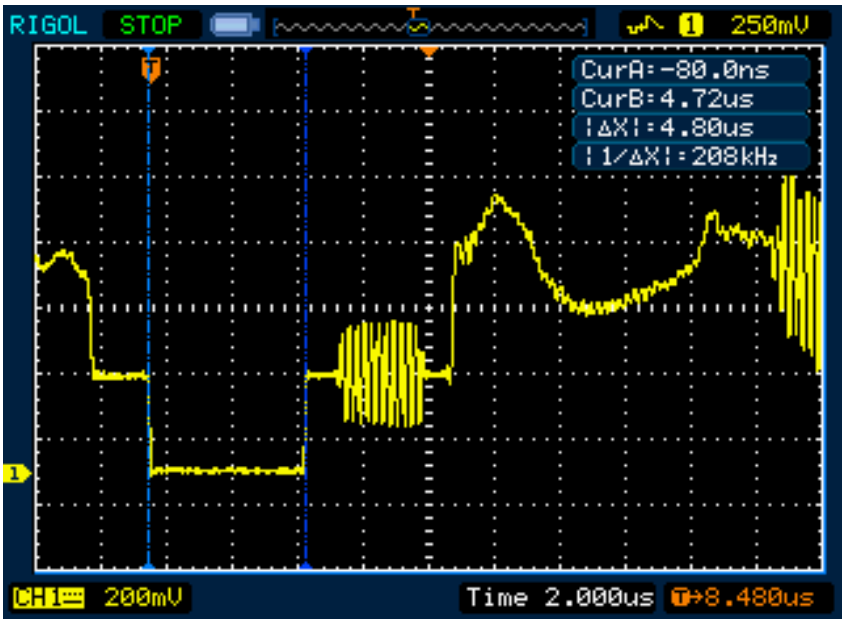
XI - Partie Physique : Analyse de trame Vidéo :

Afin de mieux comprendre l'affichage de la vidéo, j'ai choisi de relever la trame vidéo.

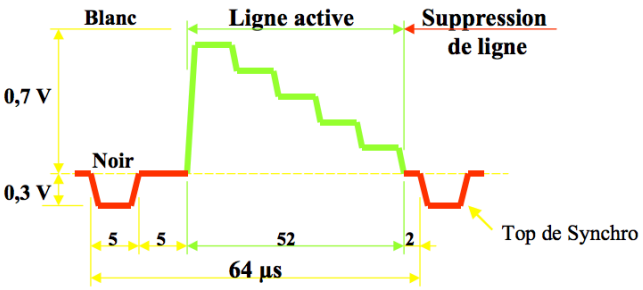


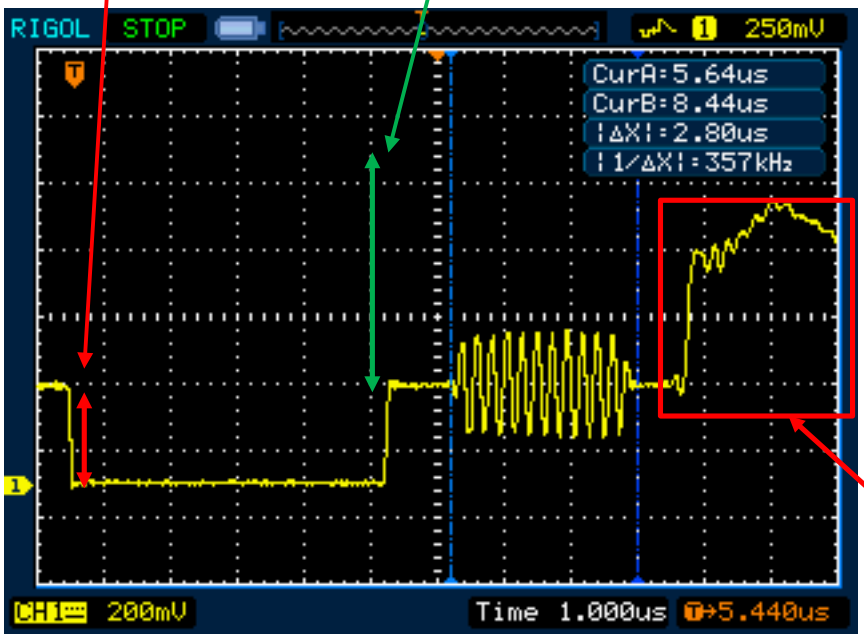
On peut voir ici la trame complète de la vidéo. Celle-ci dure environ 63,3µs ce qui correspond à 15,7kHz. Ce qui correspond aux normes car on doit trouver aux alentours de 64µs.

Ici le « synchronisation tip level » qui permet de déclencher le balayage horizontal sur l'écran. C'est donc la référence de position de la ligne horizontale sur l'écran.



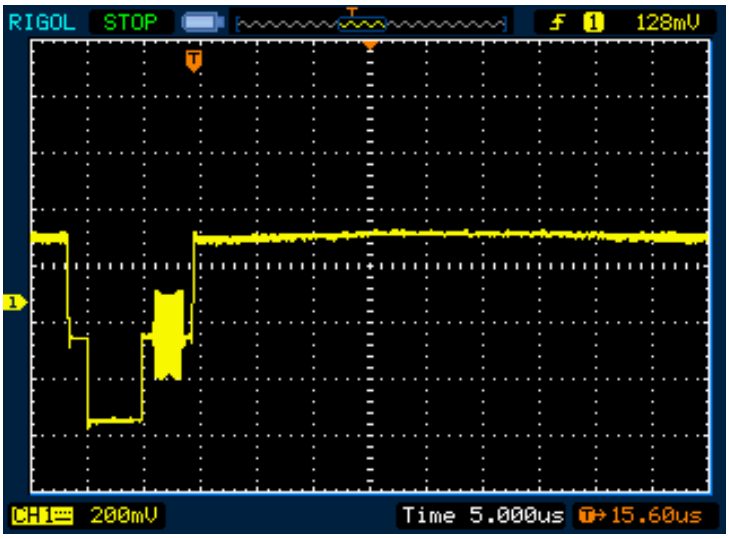
On retrouve bien la zone de noir qui fait 0,3V et la zone de blanc 0,7V.



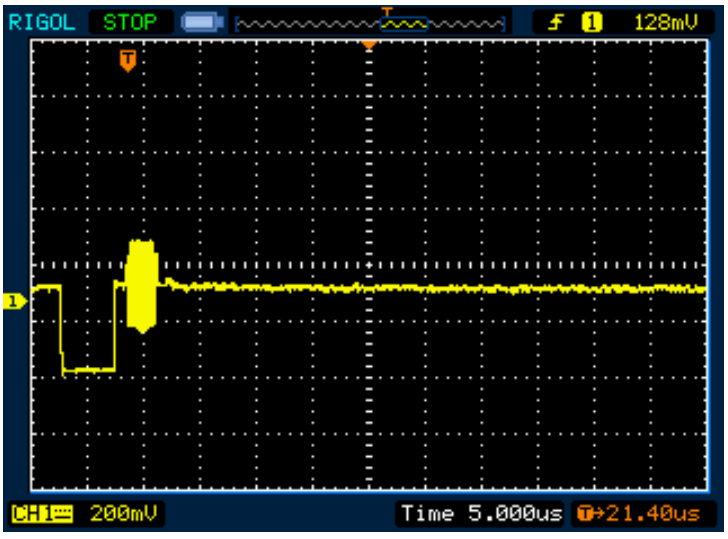


La salve suivante sert encore à la synchronisation. En effet la salve de chroma permet d'identifier le signal comme étant du NTSC.

Début du contenu de l'image

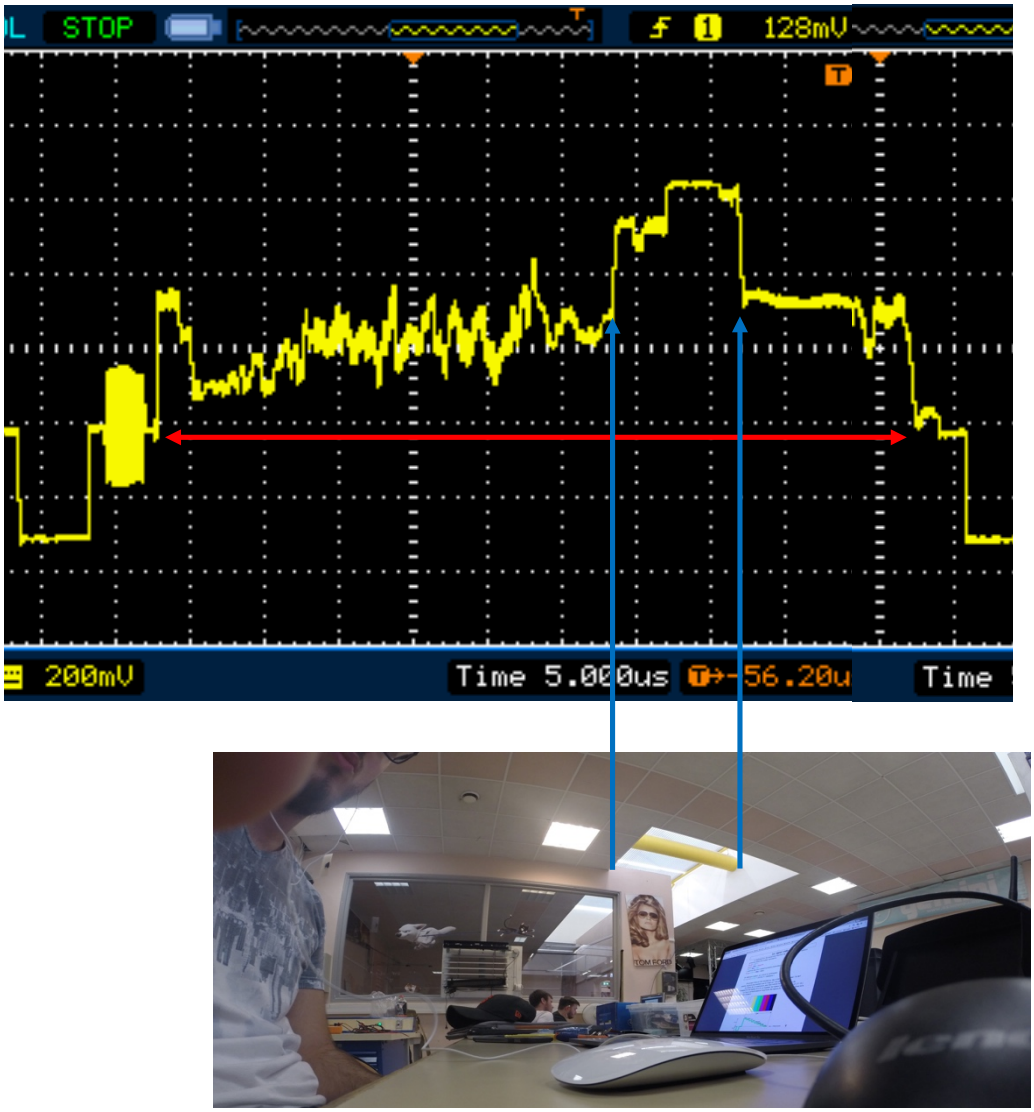


On a ici la courbe lorsque que l'on prend une photo sur fond **blanc.** On peut voir la tension maximum à 0,7V.

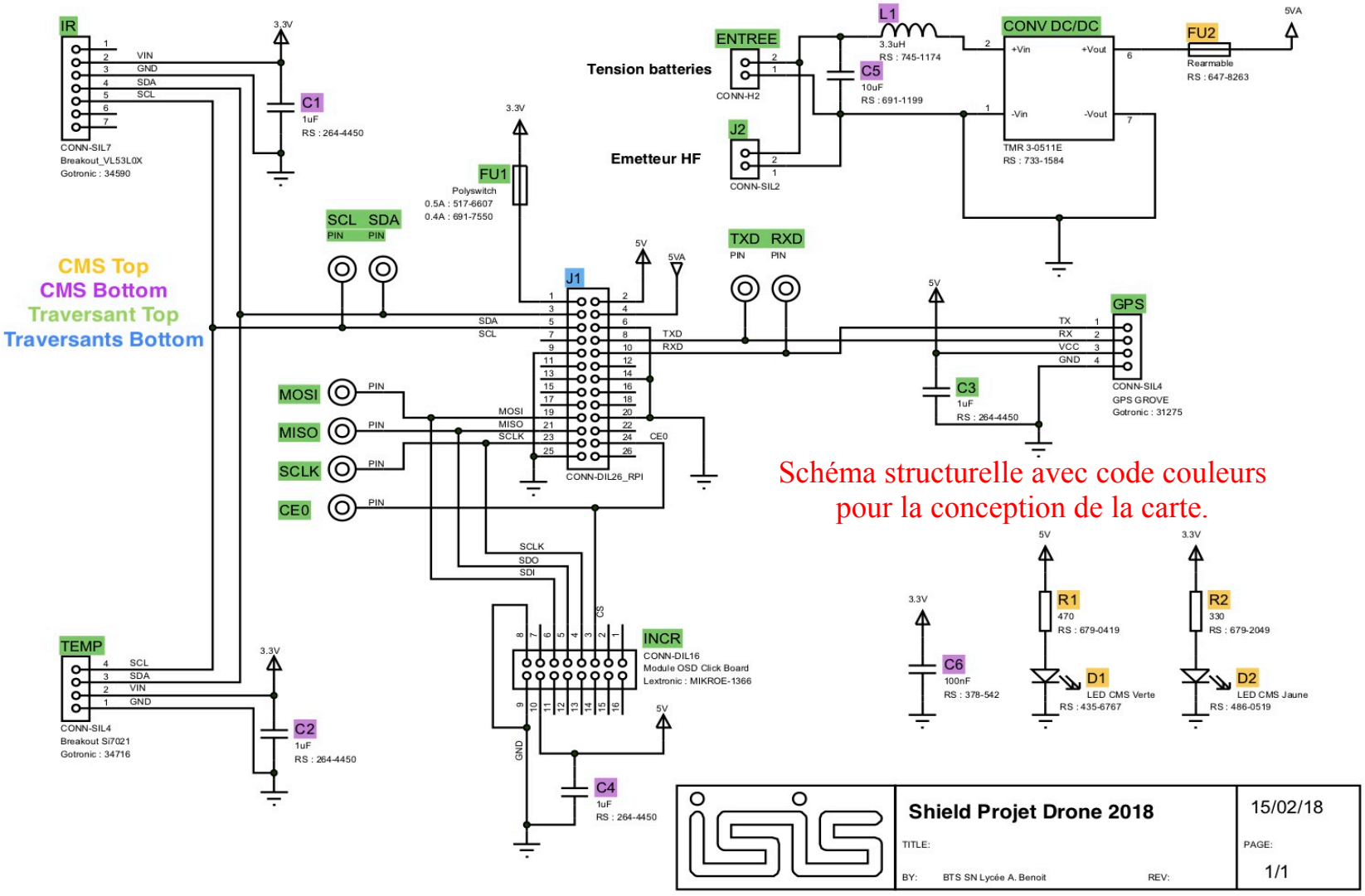


Ici la courbe lorsque que l'on prend une photo sur fond **noir.** On peut voir la tension rester au minimum soit 0V.

On a ici la trame vidéo complète dans laquelle on peut voir les différents niveaux de couleur. Dans le système NTSC les signaux R-Y et B-Y sont entrelacés il est donc assez difficile de le décoder à l'œil. On peut cependant l'assimiler à la photographie.



XII - Annexes 1 - Schéma Structurelle



XII - Annexes 2 - Nomenclatures Drone

Repère	Désignation du matériel	Valeur (Référence)	Tol.±	Equivalent	Fabricant @ Fournisseur	Boîtier	Recommandation câblage	Qté	Condition	ref rapide "RS" {FA}	Prix U HT	Prix T HT
R1	Résistance CMS	470 Ohms	1 %		Radiospares	603		1		679-0419	0,019	0,019
R2	Résistance CMS	330 Ohms	1 %		Radiospares	1206		1		679-2019	0,014	0,014
C1 à C4	Condensateur CMS	1 µF	80 %		Radiospares	805		4		264-4450	0,052	0,208
C5	Condensateur CMS	10 µF	10 %		Radiospares	1206		1		691-1199	0,199	0,199
C6	Condensateur CMS	100 nF	20 %		Radiospares	805		1		378-542	0,078	0,078
D1	LED CMS Verte	HSMG-C170			Radiospares	2012		1		435-6767	0,132	0,132
D2	LED CMS Jaune	HSMY-C170			**Radiospares**	2012		1		486-0519	0,137	0,137
FU1	Polyswitch fusible réarmable	500mA – 0,25A			Radiospares			1		517-6607	0,333	0,333
FU2	Fusible CMS réarmable	4A – 2A			Radiospares			1		647-8263	0,417	0,417
L1	Inductance de choc	3,3µH	20 %		Radiospares			1		745-1174	2,200	2,200
PIN Mâle	2 Connecteur Mâle traversant	2,54mm			Radiospares			4		896-7620	0,054	0,216
PIN Femelle	Embase femelle 20 contacts	2,54mm			Radiospares			2		600-7748	1,214	2,428
IR	Capteur de distance				**Gotronic**			1		34590	11,290	11,290
TEMP	Capteur d'humidité et de température				Gotronic			1		34716	6,880	6,880
INCR	Module «osd click board »				Lextronic			1		MIKROE-1366	32,320	32,320
GPS	Module GPS Grove 113020003				**Gotronic**			1		31275	27,420	27,420
CONV DC/DC	Convertisseur DC/DC				**Radiospares**			1		733-1584	15,650	15,650
J1												0,000
J2	Embrase 2 contacts	2,54mm			Radiospares			1		423-2914	0,260	0,260
ENTREE	Bornier 2 contacts	3,5mm			Radiospares			1		710-0444	0,520	0,520
Port Groove	Embrase CI 4 contact	2mm			Radiospares			1		512-8608	0,401	0,401
										TOTAL HT FEUILLE		101,122
										TOTAL HT GLOBAL		101,122
										COEFFICIENT		0,000

	Nom	Date	Folio
Crée :	BUFFE Jordan	05/04/2018	**1/1**
Mis à jour :	BUFFE Jordan	11/05/2018	

XII - Annexes 3 - Réalisation de la carte :

1) Se munir de la carte vierge et identifier les différents composants ainsi que les différentes couches (Top / Bottom)

2) Appliquer la pâte à braser sur les empâtements des composants CMS, sur la couche Top.

3) Positionner les composants CMS de la couche Top. Attention aux polarités des LEDs.

4) Passer la carte au four à refusions. Avec le Programme 2 (Préchauffage 90° - 2min30 montée à 220° - 1min de refusions à 250° - 1min50 refroidissement)

5) Faire les étapes 2 à 4 pour la couche Bottom.

6) Une fois la carte refroidie, se munir des composants traversant

7) Souder les composants traversant en commençant par les plus petits et les moins volumineux.

XII - Annexes 4 – Mise en service de la carte :

1) Connecter le convertisseur DC/DC sur ces broches (Faire attention au sens)

2) Mettre la carte en tension avec 7,2V et 500mA.

3) Vérifier si le 5V arrive bien grâce à la LED Verte.

4) Faire de même avec le 3,3V et la LED Rouge.

5) Vérifier avec, l'aide d'un multimètre, les tensions sur les points de tests.

6) Une fois les étapes précédentes réalisés on peut éteindre le SHIELD.

7) Ensuite connecter tous les capteurs (Capteur de Température, Distance l'Incrustateur, et le GPS).

8) Mettre en tension tous les capteurs.

9) Vérifier que tous les capteurs soient bien allumés grâce au LED présente sur chaque capteur.

10) Débranchés le SHIELD après être sûr que tous les composants fonctionnent.

11) Brancher la raspberry au SIELD ainsi qu'un écran pour voir si la raspberry n'a pas de problème au démarrage.

12) Lancé les codes et vérifier que chaque capteur fonctionne bien avec son code.

13) Eteindre la raspberry.

XII - Annexes 5 - Journal de bord :

Jeudi 11 Janvier 2018 :
Rendez-vous avec l'équipe IR EC pour une mise en place du projet.
Prise de connaissance du projet.

Vendredi 12 Janvier 2018 :
Recherche des OSD (On Screen Dislpay) pour faire le comparatif
https://www.lextronic.fr/modules-d-affichage/27494-module-osd-click-board.html
https://www.sparkfun.com/products/retired/9168
http://www.hobbytronics.co.uk/max7456-osd-breakout
https://fr.aliexpress.com/item/Free-shipping-OSD-module-Video-character-superimposition-module-Video-transparent-transmission-module-OM9024/32378935582.html

Mercredi 17 Janvier 2018 :
Rédaction du comparatif des OSD
Réalisation du diagramme de gant avec les IR

Jeudi 18 Janvier 2018 :
Mise en place du code Arduino de l'année dernière
Prise de connaissance du câblage sur le site
Câblage breadboard + Arduino + GoPro + écran

Vendredi 19 Janvier 2018 :
Prise de connaissance de la modification d'un caractère
Essai du logiciel pour changer un caractère

Mercredi 24 Janvier 2018 :
Modification sur l'OSD de l'année dernière de la table de caractères et modification du code pour ajouter température et le logo « °C »
Modification sur le deuxième OSD de la table de caractères

Jeudi 25 Janvier 2018 :
Récupération du code sous raspberry
Ajout de la bibliothèque BCM2835

Vendredi 26 Janvier 2018 :
Vérification si code fonctionnel sous raspberry : OK
Récupération du code de Mr Antoine pour tester s'il marche : pas réussi
Début de compréhension du code

Mercredi 31 Janvier 2018 :
Début de compréhension du code et de la doc technique
Prof a mis à disposition le dossier avec schéma de la carte alimentation

Jeudi 01 Février 2018 :
Mise en place du code de Mr Antoine avec QT Creator : ok
Câblage de tous les différents composant sur une seul raspberry pour voir combien la totalité consomme ≈ 500mA
Et ajout de tous les code sur cet même raspberry pour faire marcher tous les composants
Récupération du module de connexion sans fil à mettre en sortie de l'OSD et du récepteur
Il faut ajouter un RCA femelle donc recherche sur RS pour en trouver que l'on commandera
(2 séances matin en physique et aprem avec EC)

Vendredi 02 Février 2018 :
Début du routage sur Fritzing

Jeudi 08 Février 2018 :
Fin routage sur Fritzing
Ajout de ma partie incrustateur sur le ISIS
Mesure de la taille des espacements entre les 2x8 broches de l'OSD sur le routage ARES pour l'impression de la carte.

Vendredi 09 Février 2018 :
Mise en commun avec le prof du routage ISIS pour correction
Plan de mesure du capteur pour adaptation sur le routage

Jeudi 15 Février 2018 :
Professeur ajout des empâtements sur ARES.
Discussion avec tous les EC sur comment agencer les composants sur la carte
Discussion avec prof de ITEC pour conception du support de la carte sur drone

Vendredi 16 Février 2018 :
Discussion de la mise en page du rapport de projet avec les IR

21/23 Février – 14/15/16 Mars 2018 :
Rédaction revue de la 1ère revue de projet.
Début de mise en place de l'antenne RF.
Test convertisseur DC/DC

Mercredi 21 Mars 2018 :
Oral 1ère revue de projet.

22/23/28/29 Mars 2018 :
Routage du Shield pour le Drone.

Vendredi 30 Mars 2018 :
Mise en place de la communication sans fil avec le module RF

Mercredi 04 Avril 2018 :
Fabrication carte et mise en place des composants.

Jeudi 05 Avril 2018 :
Test de la carte avec les soudures pour voir si le 5V et le 3,3V arriver mais j'ai dû reprendre des soudures de l'incrustateur pour le faire fonctionner car il y semblait qu'il y avait des mauvaises masses.

Vendredi 06 Avril 2018 :
Mise en place de tous les codes sur mon Shield avec tous les capteurs : sa marche.

Mercredi 11 Avril 2018 :
Réalisation de la carte de Lozano, celui-ci c'est casser le doigt.

Jeudi 12 Avril 2018 :
Récupération de trame avec oscilloscope
Finalisation de la carte de Lozano.
Et test : sa marche.

13/18 Avril - 9/11Mai 2018 :
Aide à mes camarades EC avec leurs Shield.
Commande des carte officielles sur SeeedStudio.

Mercredi 16 Mai 2018 :
Réception des cartes.
Rédaction revue de la 2eme revue de projet.

Vendredi 18 mai 2018 :
Rédaction revue de la 2eme revue de projet.

XII - Annexes 6 - Diagramme de Gant :

		Tâches Buffe	184 h	Jeu 11/01/18	Mer 16/05/18		BUFFE Jordan
34		Recherche documentation + solution de l'année précédente	20 h	Jeu 11/01/18	Ven 19/01/18		
35		Diagramme de blocs et de séquence	10 h	Mer 24/01/18	Jeu 25/01/18	34	
36		Test du capteurs sur Arduino et Raspberry	10 h	Mer 31/01/18	Jeu 01/02/18	35	
37		Codage	30 h	Ven 02/02/18	Mer 21/02/18	36	
38		Realisation de la carte alimentation	40 h	Mer 14/03/18	Jeu 29/03/18	37	
39		Mise en place	20 h	Ven 30/03/18	Mer 11/04/18	38	
40		Routage	20 h	Jeu 12/04/18	Mer 09/05/18	39	
41		Finalisation	20 h	Jeu 10/05/18	Mer 16/05/18	40	

Diagramme de Gant prévisionnel

		Tâches Buffe	184 h	Jeu 11/01/18	Mer 16/05/18		BUFFE Jordan
34		Comparatif OSD	12 h	Jeu 11/01/18	Mer 17/01/18		
35		Recherche documentation + mise en place solution de l'année précédente	4 h	Jeu 18/01/18	Jeu 18/01/18	34	
36		Modification de la table de caractères	7 h	Ven 19/01/18	Mer 24/01/18	35	
37		Code sous QT pour raspberry	19 h	Jeu 25/01/18	Jeu 01/02/18	36	
38		Routage Fritzing	3 h	Ven 02/02/18	Ven 02/02/18	37	
39		Réalisation Schéma structurel ISIS + Début routage ARES	19 h	Jeu 08/02/18	Jeu 15/02/18	38	
40		Rédaction Revue de projet 1	23 h	Ven 16/02/18	Ven 16/03/18	39	
41		Routage ARES	19 h	Jeu 22/03/18	Jeu 29/03/18	40	
42		Mise en place module sans fil	3 h	Ven 30/03/18	Ven 30/03/18	41	
43		Réalisation de ma carte avec les composants + Test avec les différents modules	13 h?	Mer 04/04/18	Ven 06/04/18	42	
44		Réalisation carte Lozano + Test	8 h	Mer 11/04/18	Jeu 12/04/18	43	
45		Analyse d'une trame vidéo	3 h?	Ven 13/04/18	Ven 13/04/18	44	
46		Aide de mes camardes EC avec leurs Shield + Commande des cartes officielles	20 h?	Mer 18/04/18	Ven 11/05/18	45	
47		Rédaction revue de pojet 2	13 h	Mer 16/05/18	Ven 18/05/18		

Diagramme de Gant réel