

Drone 2 : D'imagerie et de Télémétrie



Académie Aix Marseille

Lycée Alphonse Benoit

Cours Victor Hugo

84800 L'Isle sur la Sorgue

BOTTIGLIERO Nicolas

TISON Sylvain

DEMONCHY Maxime

Sommaire

Présentation Générale.....	6
Introduction :	6
Cahier Des Charges :	7
Description des besoins de L'EDD :	7
Description des besoins du GCS :	8
Spécification :	9
PROJET DRONE : TISON SYLVAIN BTS SYSTÈMES	
NUMÉRIQUES.....	10
I – EC:MESURER (GÉOLOCALISATION ET ALTITUDE)	10
Présentation personnelle :	11
Mise en œuvre du module GPS :	12
Schéma structurel du module GPS :	13
Essais par câblage Rapide :	14
Analyse de Trame :	15
Trame GPGGA :	15
Problèmes survenus :	16
UART :	17
Saleae :	17
Conception du schéma structurel	19
GPS :	20
Capteur de distance : BOTTIGLIERO Nicolas	20
Incrustateur Vidéo : BUFJE Jordan.....	21
Capteur d'humidité : LOZANO Dorian.....	22
Bloc Alimentation :	22
Autres Composants :	23
Création d'un support supplémentaire pour le drone.....	24
Routage de la carte d'extension	25
GPS :	25
Capteur de Distance et de Température :	26
Incrustateur :	27

Bloc Alimentation :	27
Test de la carte d'extension	28
Fabrication :	28
Tests à effectuer en cas de dysfonctionnement : Processus de maintenance curative	28
Création du Shield final.....	29
Test et mise en place finale :	30
Annexe 0 : Gant et Journal de bord.....	31
Gant Final :	31
1. Journal de bord.	32
2. Journal de bord.	33
Annexe 1 : Code du module GPS	34
Annexe 2 : Capture de la trame NMEA avec l'analyseur logique Saleae :	35
Annexe 3 : Schéma Structurel.....	36
Annexe 4 : Support du drone.....	37
Annexe 5 : Premier schéma de routage : 28/03	38
Annexe 5 : Schéma de routage final : 13/04	39
Annexe 6 : Essai du PCB de test avec tous les capteurs	40
Annexe 7 : Nomenclature.....	41
PROJET DRONE : BOTTIGLIERO NICOLAS : BTS SYSTÈMES NUMÉRIQUES	42
II – EC:MESURER (DISTANCE)	42
Introduction :	43
Capteur de distance infrarouge VL53L0X :	44
Schéma structurelle du VL53L0X :	45
Infrarouge :	46
Schéma de câblage rapide :	47
Mise en œuvre de l'exemple avec Cmake sur Rpi :	48
Programme de test :	49
Programme de base :	49
Programme modifié :	50
Problème rencontré	51

Lecture de trame	51
Conceptions du circuit imprimé :	52
Schéma structurel :	52
Les capteurs :	53
Convertisseur DC/DC :	54
Routage sous ARES :	54
Etapes de conception de la carte test :	56
Analyse de trame :	60
Création de la plaque :	61
Annexes 1 :	63
Journal de bord :	63
Annexes 2 :	65
Schéma ISIS :	65
Annexes 3 :	66
Gantt :	66
Annexe 4 :	67
Diagramme de bloc :	67
Annexe 5 :	68
Conception de la carte :	68
Annexe 6 :	69
En cas de problème :	69
Si la communication I2C ne fonctionne pas :	69
Si la LED verte ne s'allume pas :	69
Si la raspberry reboot au démarrage :	69
Annexes 7 :	70
Nomenclature :	70
PROJET DRONE: DEMONCHY Maxime BTS SYSTEMES NUMERIQUES	71
I. IR : DEVELOPPEMENT DE L'EQUIPEMENT DU DRONE	71
Présentation personnelle	72
Conception des multithreads	73
Conceptions de la zone de mémoire partagé	73

Diagramme de Classe.....	74
Connexion entre la Raspberry et la GoPro.....	75
Présentation du WiFi de la Raspberry.....	76
Problème rencontré	77
Semaine à venir :	77

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'huissières, 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 :

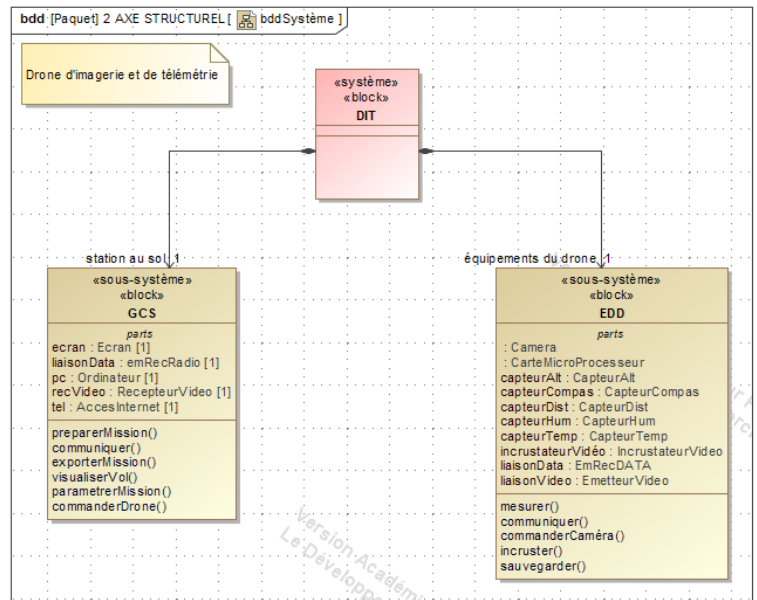
Le système comprend deux parties :

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.

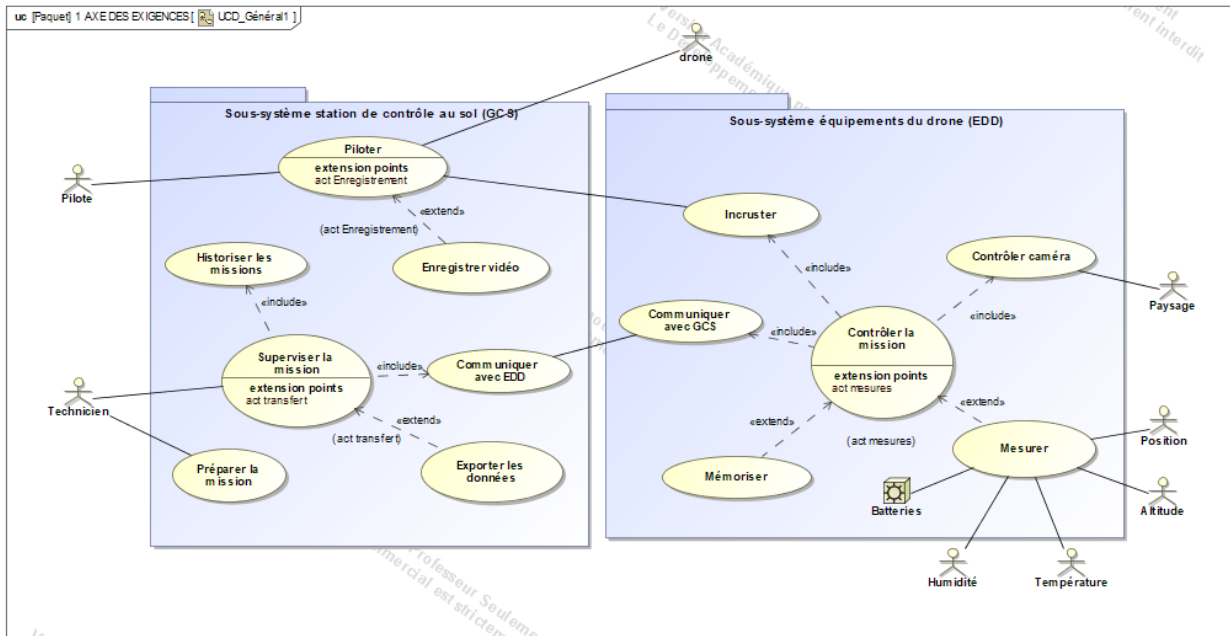
- D'effectuer l'acquisition des mesures en temps réel.

- 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 :



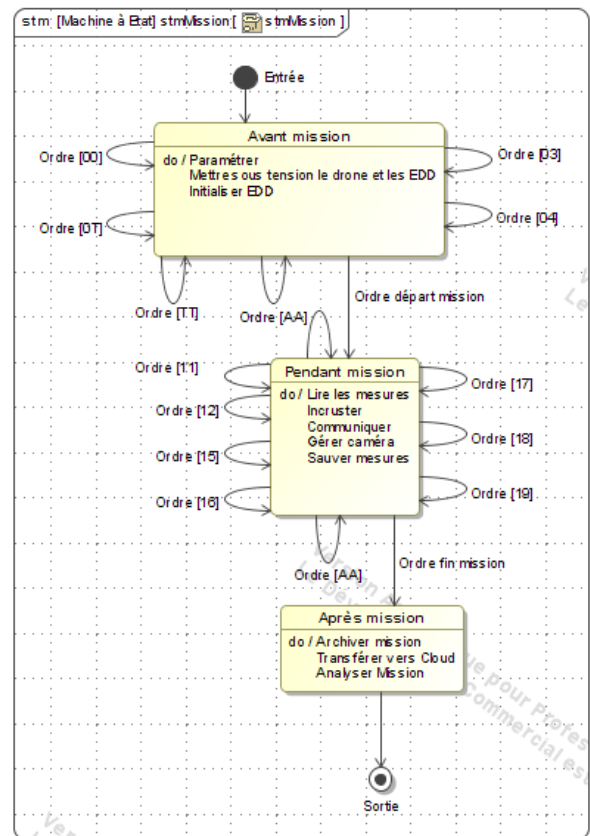
Durant son fonctionnement, le système peut se trouver dans un des trois états suivants :

- 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.



PROJET DRONE : TISON SYLVAIN **BTS SYSTÈMES NUMÉRIQUES**



I – EC:MESURER (GÉOLOCALISATION ET ALTITUDE)

Présentation personnelle :

Dans le cadre de ce projet et avec la collaboration de 6 étudiants (3 Électroniciens et 3 Informaticiens), mon objectif est de mettre en œuvre une carte de géolocalisation, ainsi que de développer une application permettant de vérifier le bon fonctionnement de cette même carte. Par la suite je devrais concevoir/réaliser/tester une nouvelle carte d'extension pour Raspberry Pi (shield) intégrant tous les breakouts des capteurs et autres.

La création/réalisation du shield sera un travail commun avec les autres électroniciens de ce projet, seul le routage sera fait personnellement.

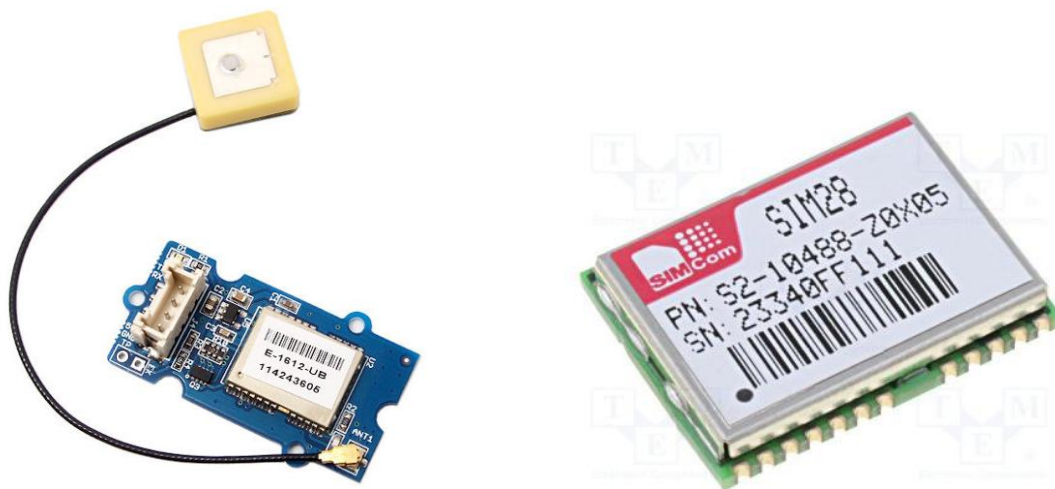
Durant toute la mise en œuvre de ce projet, il sera important de trouver des solutions cohérentes qui ne perturberont pas le vol du drone.

Le journal de bord et le gant sont disponible en annexe 0 afin de pouvoir observer chronologiquement les tâches effectuées.

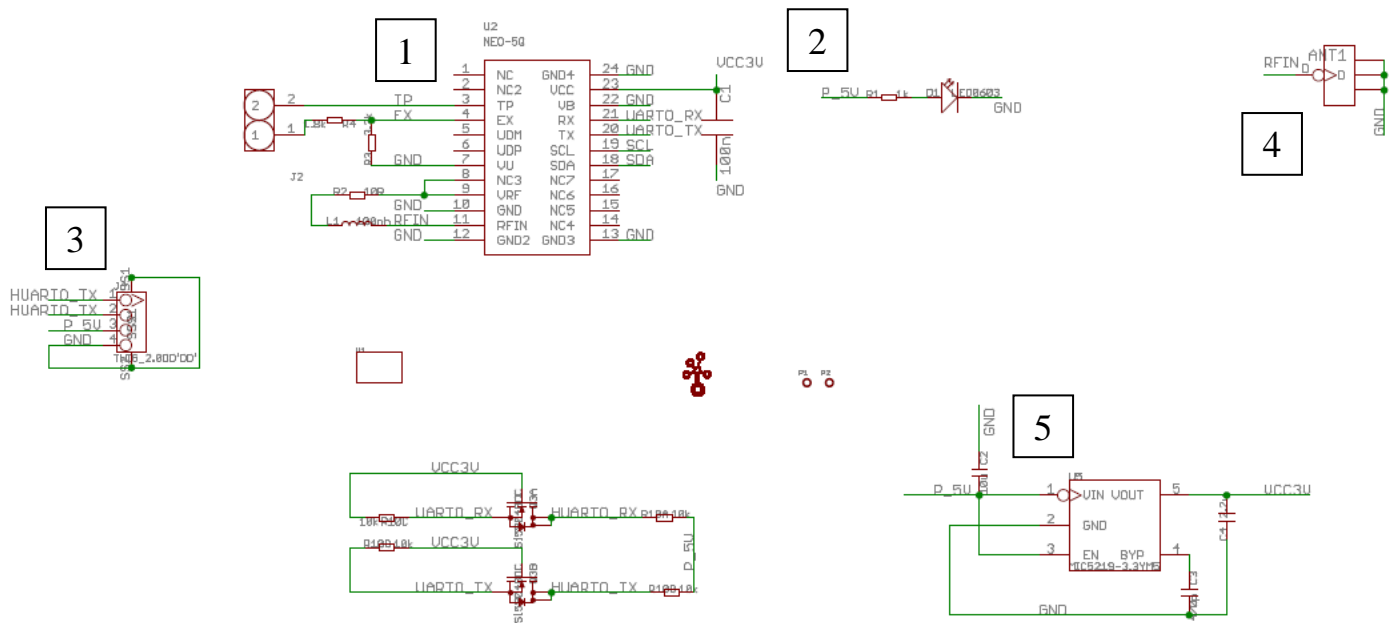
Étudiant	Liste des fonctions assurées par l'étudiant	Installation :
EC3	<p>EDD : UC Mesurer (Géolocalisation, altitude et distance)</p> <p>Valider une nouvelle solution.</p> <p>Concevoir/Réaliser/Tester une nouvelle carte d'extension pour Raspberry Pi intégrant tous les breakout des capteurs et autres.</p> <p>Développer une application permettant de vérifier, le bon fonctionnement de la géolocalisation.</p>	<p>- Rpi : bus UART et librairie C/C++ BCM2835.</p> <p><i>Mise en œuvre :</i></p> <p>- Valider par prototypage rapide le module GPS grove.</p> <p><i>Réalisation :</i></p> <p>- Participer avec les autres étudiants EC à la conception d'un schéma commun intégrant tous les composants de l'EDD.</p> <p>- Concevoir une carte électronique personnelle compatible avec tous les contrats EC.</p> <p><i>Documentation :</i></p> <p>- Bibliothèque C/C++ d'accès au module GPS.</p> <p>- Documents de fabrication de la carte. Ces documents devront permettre une fabrication industrielle du circuit imprimé.</p>

Mise en œuvre du module GPS :

La première partie de mon projet consiste à récupérer et analyser des données reçues par le module GPS Grove 113020003. Ce module permet de connaître notre position (longitude, latitude et altitude) en temps réel grâce à la triangulation de signaux électromagnétiques émis par satellites. La communication se fait via un port série et utilise donc l'UART pour communiquer.



Il a été choisi de mettre en œuvre ce module GPS de part son faible coût (cf nomenclature) et sa disponibilité, il est également utilisé dans d'autres projets par les enseignants et possède également un adaptateur de tension ceux qui sera utile lors des tests sur Rpi.

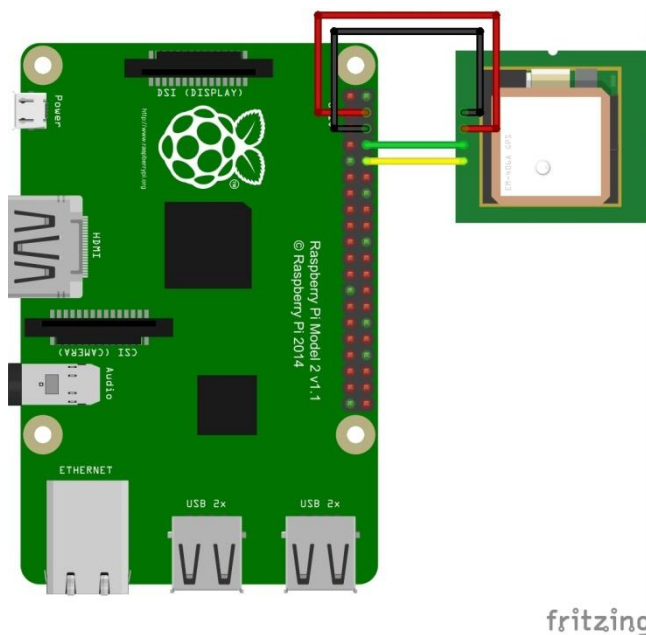
Schéma structurel du module GPS :

Nous pouvons voir le schéma de la SIM28 (1), ce composant est celui qui permet de traiter les coordonnées GPS et de les restituer. RFIN correspond à la connexion à l'antenne radio, RX et TX sont les broches de communication vers le port grove, sa tension d'alimentation en VCC doit être de 3.3V. Sur la droite nous avons la diode (2) qui nous signalera de l'alimentation ou non du module.

Sur la gauche nous avons le connecteur grove (3) qui permettra de communiquer avec la Raspberry Pi, il comporte deux broches RX et TX pour communiquer, une broche d'alimentation et une masse.

Sur la droite le connecteur de l'antenne radio (4) qui réceptionnera les données satellites reçues et les retransmettra au SIM28.

Cette structure (5) (level shifter) adaptera le niveau de tension de la Rpi (5V ou 3.3V) pour obtenir les 3.3V désirés.

Essais par câblage Rapide :

Le module GPS se raccordant sur un port du Grove Base Shield, il a été nécessaire d'adapter les branchements lors des tests. Ci-dessus nous pouvons voir le schéma de câblage rapide réalisé sur Fritzing et une photo du câblage réel lors des phases de test.

Pour pouvoir lire les données reçues par le GPS, j'ai dû récupérer un code en C++ utilisé plus tôt dans l'année lors d'un TP. Ce code ne fait que lire les données transmises par le module GPS (cf annexe 1).

Analyse de Trame :

Trame Reçue :

```
pi@raspberrypi:~/Projet $ ./GPS1
$GPGGA,152039.000,4355.2786,N,00502.8065,E,1,6,1.41,48.5,M,49.1,M,,*67
$GPGSA,A,3,15,12,24,28,20,13,,,,,1.66,1.41,0.87*03
$GPGSV,3,1,10,15,78,247,17,13,60,132,19,24,45,284,14,17,31,096,*7D
$GPGSV,3,2,10,20,27,232,12,28,26,046,16,19,25,122,18,12,19,215,14*7A
$GPGSV,3,3,10,10,10,325,13,05,03,184,*75
$GPRMC,152039.000,A,4355.2786,N,00502.8065,E,1.53,192.37,240118,,A*65
$GPGGA,152040.000,435
```

Ci-dessus nous pouvons voir les trames transmises par le GPS, il est maintenant nécessaire de les analyser afin de connaître le positionnement de notre capteur.

Trame GPGGA :

La trame GPGGA est très couramment utilisée puisqu'elle nous permet d'acquérir toutes les données nécessaires à la géolocalisation complète du capteur. Chaque trame commence par le caractère « \$ », et chaque champ de données est séparé par le caractère « , ».

```
pi@raspberrypi:~/Projet $ ./GPS1
$GPGGA,152039.000,
```

\$GPGGA	Caractère 1 → 6	Trame GPS de type GGA
152039.000	Caractère 8 → 17	Trame envoyée à 15h 20m 39.000s

Lors des tests, l'heure reçue est retardée de 1h, cela est sûrement dû à un changement d'heure été/hiver.

```
, 4355.2786, N, 00502.8065, E,
```

4355.2786, N	Caractère 19 → 29	Latitude 43°55'27.86 Nord
00502.8065, E	Caractère 31 → 42	Longitude 5°02'80.65 Est

Les valeurs de la latitude et de la longitude sont correctes et correspondent à la localisation du capteur lors des tests (l'Isle sur la sorgue).*

1 , 6 , 1 . 4 1 , 6 1 . 5 , M , 4 9 . 1 , M , , * 6 7

1	Caractère 44	Type de positionnement (1 = GPS)
6	Caractère 46	Satellites calculant les coordonnées
1.41	Caractère 48 → 51	Précision horizontale 1.41
61.5, M	Caractère 53 → 58	Altitude en mètres 61.5m
*67	Caractère 68 → 70	Contrôle de parité

La précision horizontale est comprise entre 1 et 3, ce qui signifie que la mesure est très précise. Plus la valeur est élevée, moins la mesure est précise, au-delà de 6 la mesure est trop peu fiable pour être acceptable. L'altitude ici est de 61.5m, cela est cohérent avec notre localisation.

Problèmes survenus :

Lors des tests et de la mise en route du module GPS, une difficulté lors de la réception de données est survenue. Si la position du GPS et de l'antenne restent stable le problème ne survient pas. Mais si le GPS est déplacé vers un autre endroit, les données sont partiellement transmises et deviennent illisibles. Il faut par la suite attendre que les données soient de nouveau réceptionnées par le module (entre 15mn et 1h d'attente).

Ce problème reste cependant très anecdotique, car lors de tests en extérieur cela ne se produit plus. L'erreur provient donc du GPS qui ne peut réceptionner correctement en intérieur.

```
pi@raspberrypi:~/Projet $ ./GPS1
$ GPGGA,15.2426,0.86,,,,,0,0,,,M,,M,,*40
$ GPGSA,A,1,,,,,,,*1E
$ GPGSV,1,1,01,28,,,21*71
$ GPRMC,15.2426,0.86,V,,,,,0.00,0.00,180.118,,,,N*44
$ GPGGA,15.2427,0.86,,,,,0,0,,,M,,M,,*41
$ GPGSA,A,1,,,,,,,*1E
```

Liens :

Analyse de trames :

http://www.catb.org/gpsd/NMEA.html#_satellite_ids

http://www.gpspassion.com/forumsen/topic.asp?TOPIC_ID=17661

https://fr.wikipedia.org/wiki/NMEA_0183

Vérification coordonnées GPS :

<https://www.coordonnees-gps.fr/>

UART :

Le capteur GPS utilise la liaison UART pour communiquer, cette liaison est asynchrone et peut être à la fois émettrice ou réceptrice (RX / TX). La trame se compose de :

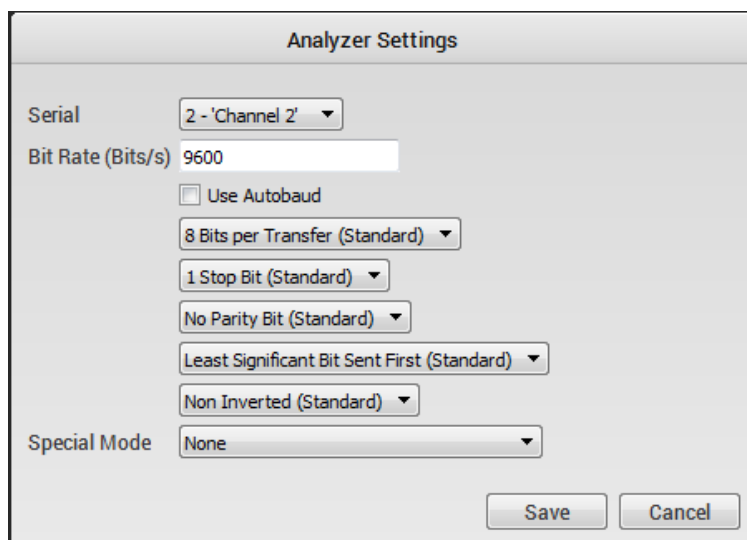
- 1 bit de start : toujours à 0
- 8 bits de données
- 1 bit de parité : optionnel
- 1 bit de stop : toujours à 1

Les niveaux de tension sont de type TTL (0V → état bas : 3.3V → état haut), le niveau de repos logique est à l'état haut (cela permet de ne pas le confondre avec une absence de tension) et lors de l'envoi de données le LSB est transmis en premier.

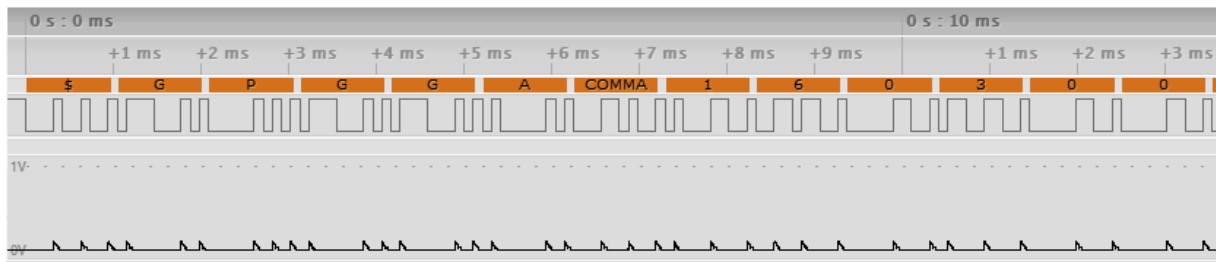
Saleae :

Pour récupérer une trame UART avec saleae, je l'ai raccordé à la broche TX du GPS (les données transitent du capteur vers la raspberry) et à la masse.

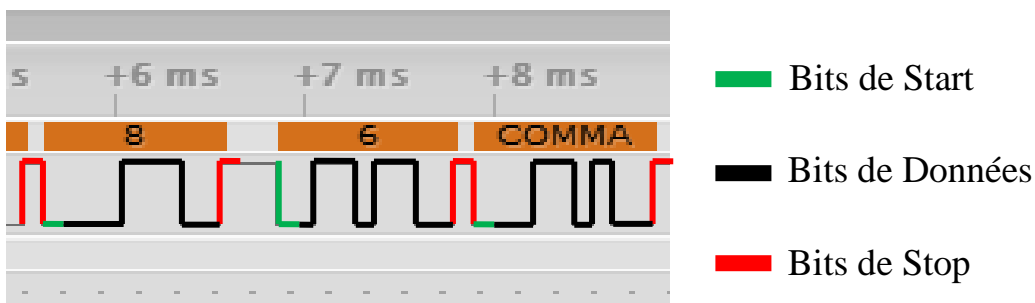
Voir annexe 2.



Les options de l'analyseur ont été calibrées pour une liaison UART.



L'analyseur lit directement la trame transmise avec les bons réglages, cela correspond à un début de trame GPGGA envoyée à 16h03.



Si l'on analyse plus en détails les données reçues pour les trois octets ci-dessus (les caractères sont codés en ASCII) :

Premier octet « 8 » :

LSB 00011100 MSB → MSB 00111000 LSB → 0x38 → caractère « 8 »

Second octet « 6 » :

LSB 01101100 MSB → MSB 00110110 LSB → 0x36 → caractère « 6 »

Troisième octet « COMMA » :

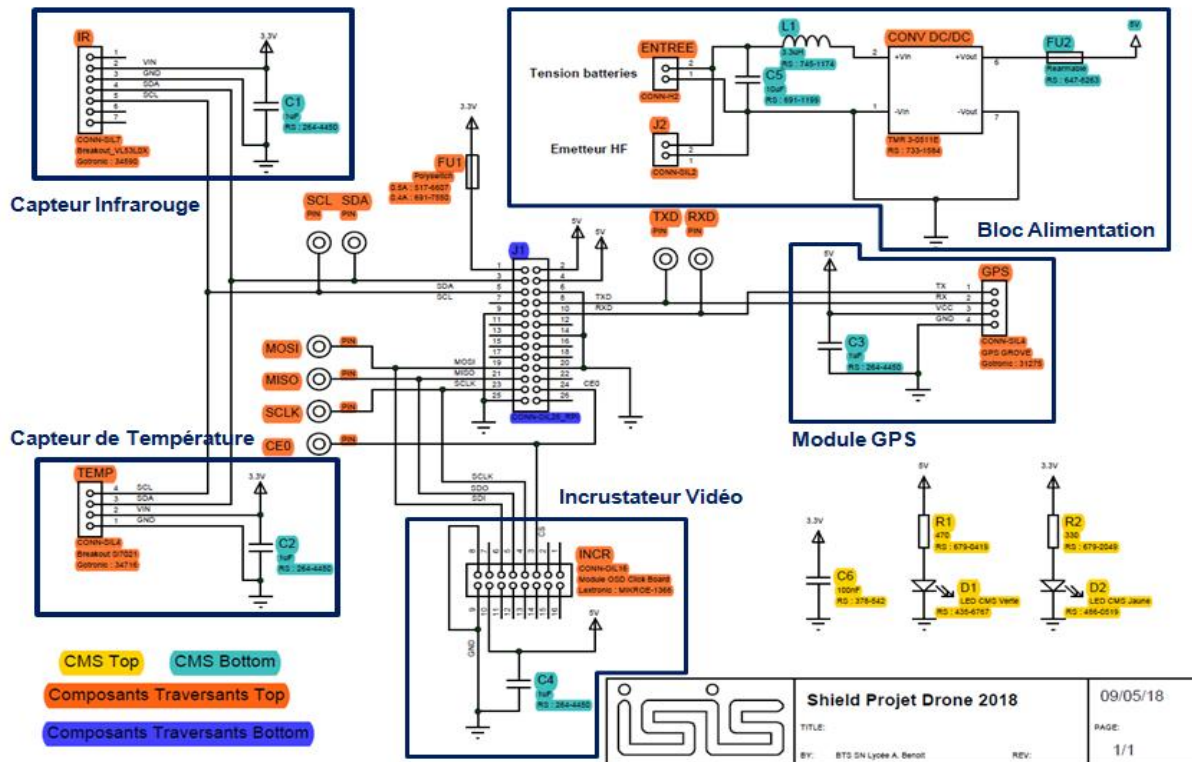
LSB 00110100 MSB → MSB 00101100 LSB → 0x2C → caractère « , »

Dans le cas présent la vitesse de transmission est de 9600 bits/s, il est donc possible de connaître la durée de transmission d'un caractère : 10bits → 1.04ms. Ainsi que la durée de transmission d'une trame entière : 70 caractères → 72.8ms.

Conception du schéma structurel

Le schéma structurel complet est disponible en Annexe 3.

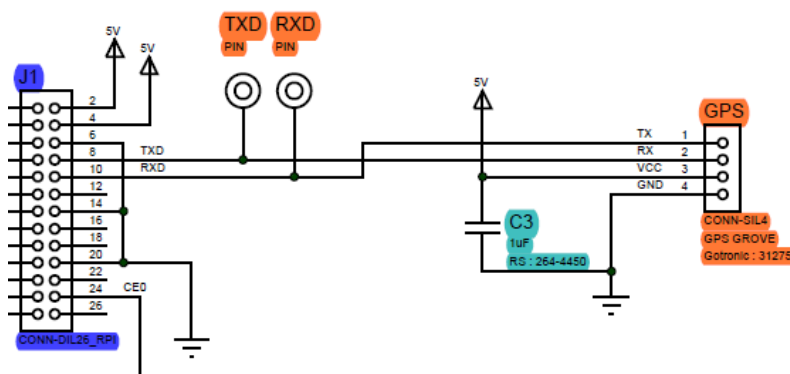
La validation du fonctionnement du module GPS grove étant effectuée, ainsi que celle des capteurs de température, d'humidité et de l'incrustateur vidéo. L'étape suivante est la réalisation d'une nouvelle carte d'extension intégrant les modules cités ci-dessus, pour cela, avec les autres étudiants EC nous avons conçu le schéma structurel suivant.



GPS :

Le GPS est disposé à proximité des broches RX et TX, il sera alimenté en 5V par le gpio de la Raspberry Pi, deux pins test sont positionnés afin de pouvoir analyser la trame facilement, et un condensateur de 1 μ F est placé afin de servir de réservoir d'énergie.

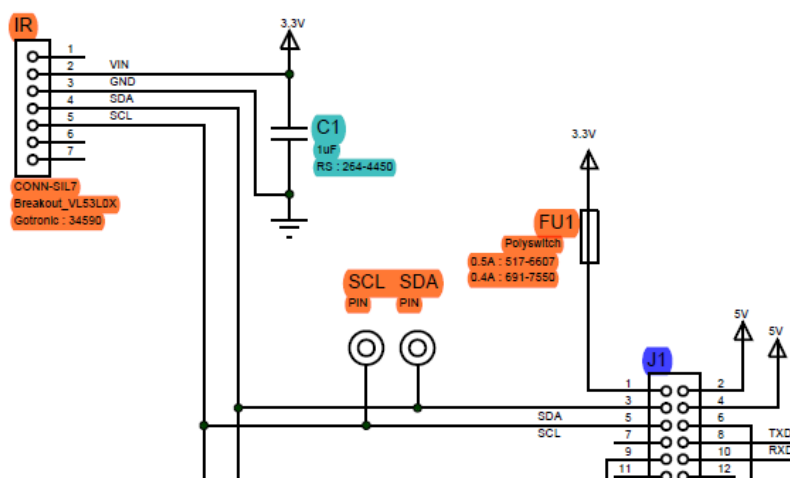
Le connecteur du GPS et les pins de test seront placées sur le coté top du shield afin de faciliter leur accès, le condensateur sera placé du coté bottom pour des facilités de routage.



Capteur de distance : BOTTIGLIERO Nicolas

Le capteur de distance infrarouge sera relié aux broches SDA et SCL, nous l'alimenterons avec 3.3V et lui aussi sera relié à un condensateur de 1 μ F et 2 pin test. Le polyswitch FU1 est utilisé comme protection pour le 3.3V.

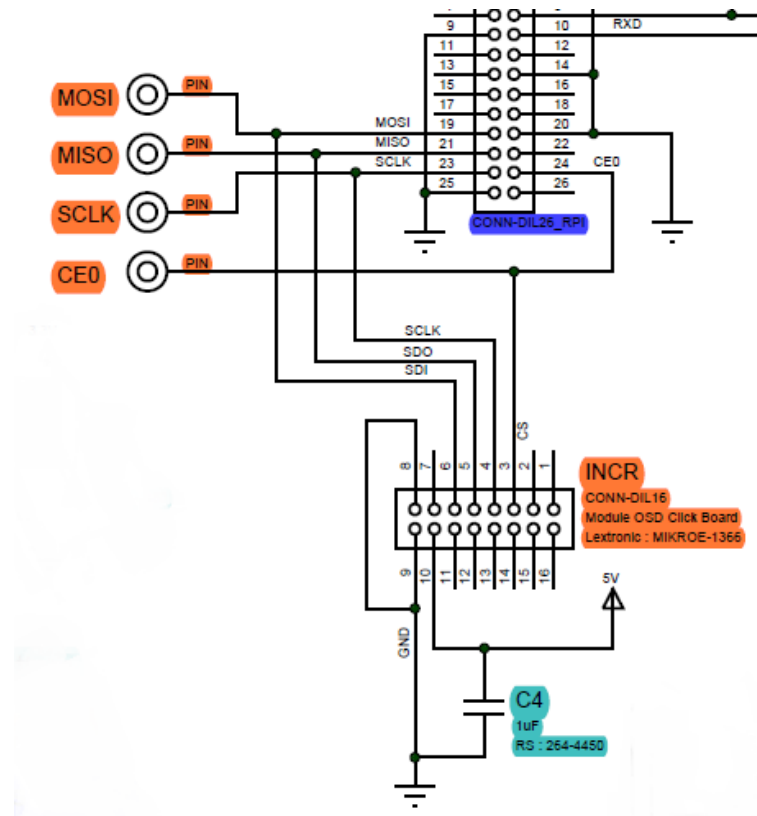
Le capteur infrarouge et les pins test sont eux aussi placés sur le coté top afin de faciliter leur accès. Le condensateur (bottom) et le polyswitch (top) seront positionnés ainsi pour des facilités de routage.



Incrustateur Vidéo : BUFJE Jordan

L'incrustateur communiquant en SPI, il sera donc relié aux broches MOSI, MISO, SCLK et CE0. Les pins permettront d'analyser la trame facilement et le condensateur stabilisera son alimentation.

Leurs dispositions suivent la même logique que pour les composants précédents, les pins et le capteur seront sur le côté top tandis que le condensateur sera du côté bottom.

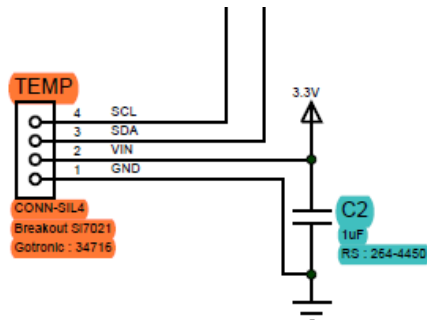


Le connecteur 26 broches sera placé sur la couche bottom afin de pouvoir se fixer directement sur la Raspberry Pi.

Capteur d'humidité : LOZANO Dorian

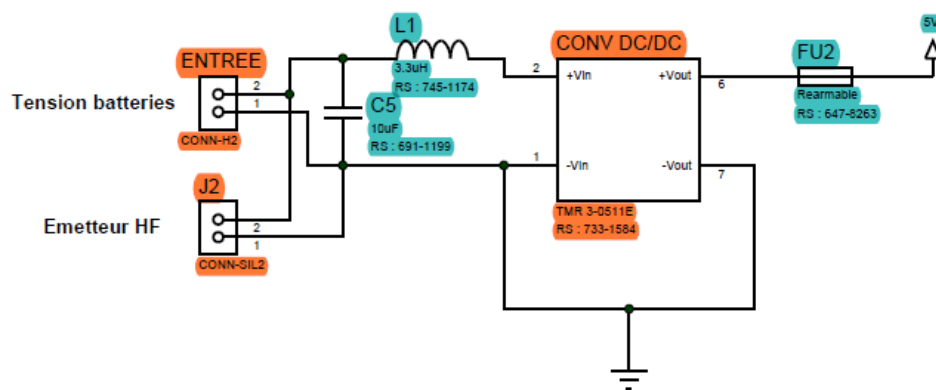
Le capteur d'humidité sera également relié aux broches SCL et SDA pour communiquer, son adresse étant différente du capteur de distance aucun conflit ne sera présent lors du transfert de données.

Comme les autres capteurs, il sera positionné sur le coté top du shield. Le condensateur aura la même fonction que pour les autres capteurs.



Bloc Alimentation :

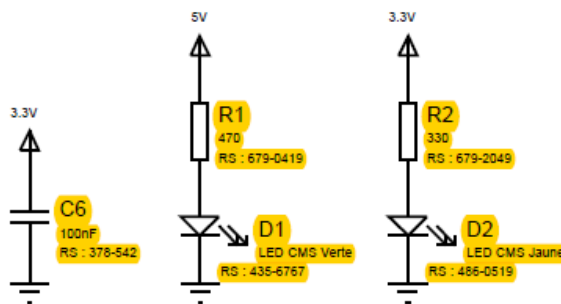
Lors des phases de vol, les batteries du drone devront alimenter le shield et la Rpi, l'alimentation de la Rpi se fera grâce à une des broches 5V du gpio. La tension sortant des batteries étant de 7V, il est nécessaire d'adapter la tension pour avoir le 5V en entrée de la Rpi. Les connecteurs ENTREE et J2 permettront de connecter respectivement les batteries et l'émetteur haute fréquence.



Autres Composants :

La résistance R1 adaptera la tension et l'intensité pour la diode verte D1 qui indiquera si le 5V est bien transmis. La résistance R2 adaptera la tension et l'intensité pour la diode jaune D2 qui indiquera si le 3.3V est bien transmis. C6 est un condensateur de découplage.

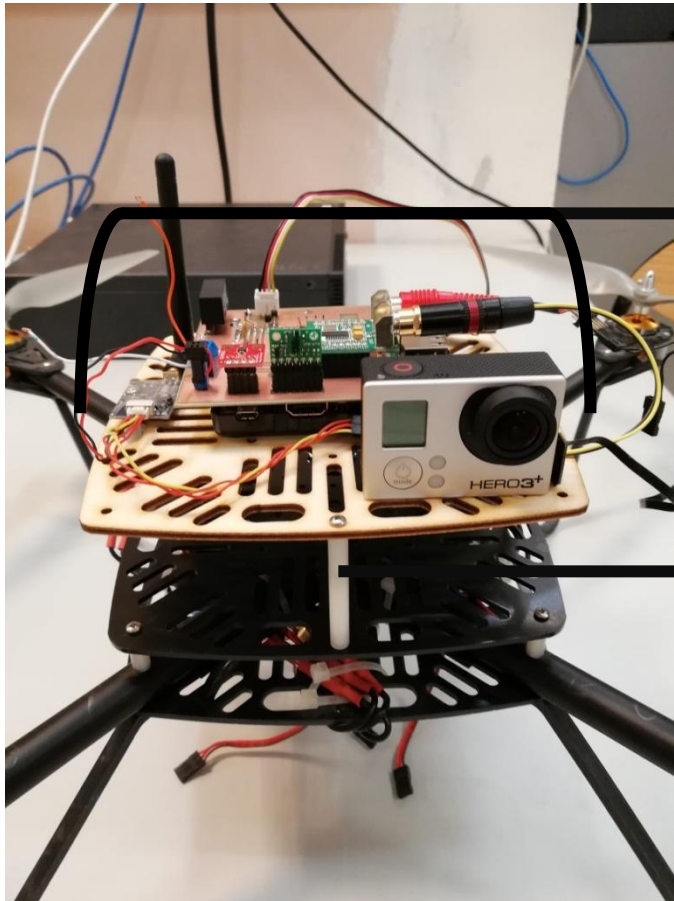
Ces composants et plus particulièrement les diodes ont été positionné sur le coté top afin d'être plus visibles et d'identifier immédiatement si les sources d'alimentation sont bien présentes.



Création d'un support supplémentaire pour le drone

Le support est visible en Annexe 4

Afin de pouvoir positionner les capteurs sur le drone il a été décidé de créer une nouvelle plaque en bois, les batteries utilisant beaucoup d'espace il était impossible de positionner la raspberry sur le drone.



→ L'équipement complet viendra se fixer sur la plaque du dessus.

→ Les deux batteries prendront tout l'espace situé entre les deux plaques.

Pour pouvoir réaliser cette seconde plaque, il aura fallu scanner la première avec des proportions identiques. Ensuite il a été possible de la faire fabriqué par des élèves d'une autre section de l'établissement.

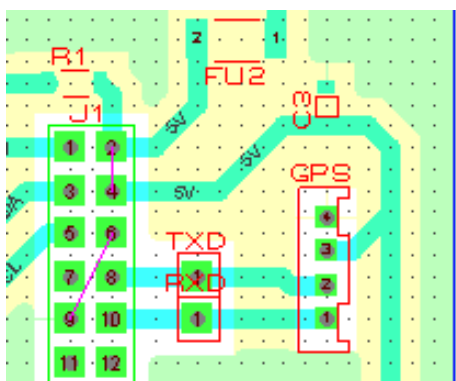
ROUTAGE DE LA CARTE D'EXTENSION

Visible en annexe 5

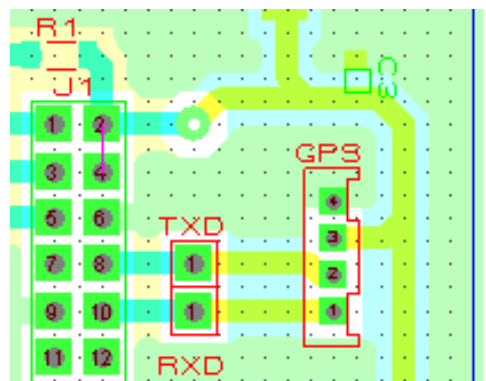
Le schéma structurel étant terminé, nous avons pu avec les autres étudiants EC commencer le schéma de routage individuellement. En annexe nous pouvons voir le premier et dernier schéma de routage réalisé. Le connecteur 26 broches a été placé vers la droite de façon à correspondre au gpio de la Raspberry, les capteurs on ensuite était placés au plus près de leurs broches de communication.

GPS :

Avant



Après

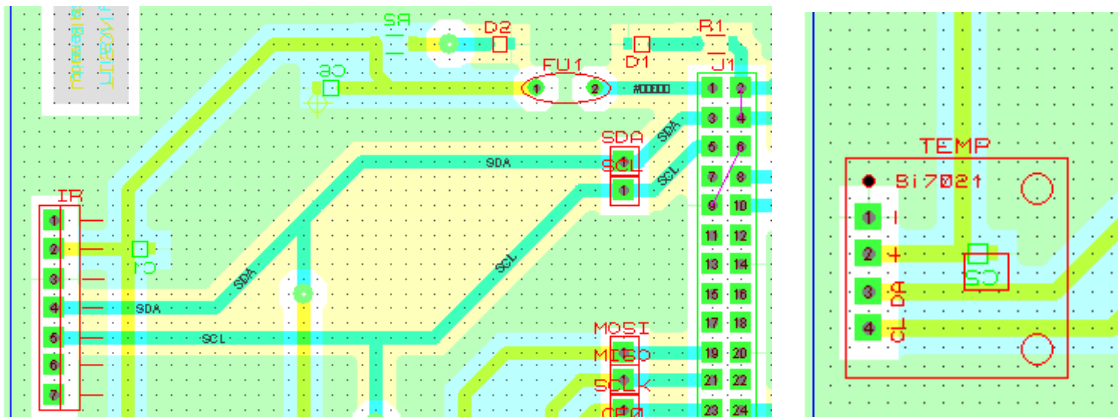


Le GPS et ses pins test sont placés juste à coté des broches RX et TX, le seul changement effectué à été de passer les pistes de la couche top à la couche bottom afin de faciliter la soudure du composant, cela permet également à la masse du gps d'être reliée à la masse de la carte.

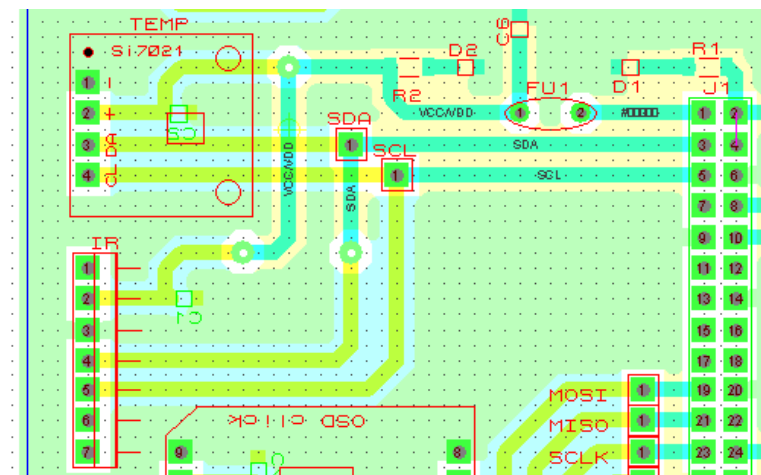
Sur la globalité de la carte, le frein thermique fût supprimé car il n'était pas utile lors de la réalisation de carte. Pour chaque capteur, les condensateurs sont placés le plus près possible de leurs broches d'alimentation afin d'avoir une efficacité optimale.

Capteur de Distance et de Température :

Avant



Après

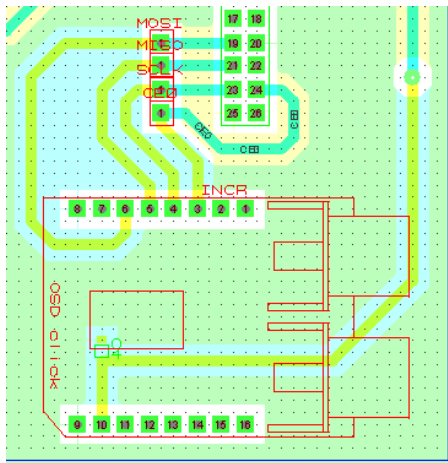


Dans les deux cas, le capteur de distance est placé vers l'avant au milieu de la carte afin de pouvoir au mieux analyser la distance, les pistes top furent passées côté bottom pour des facilités de soudage également. Le capteur de température fut déplacé sur la carte afin d'économiser de l'espace.

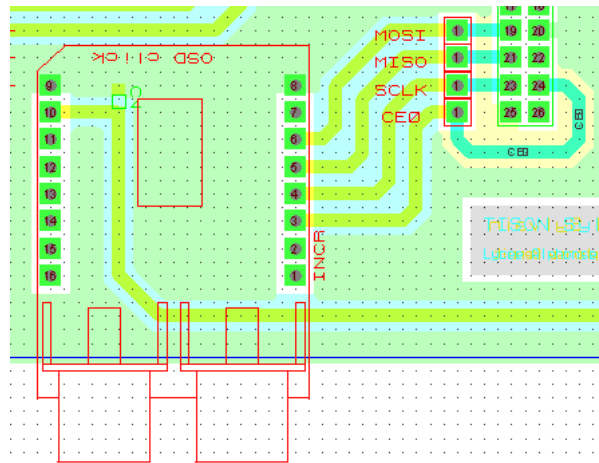
Les diodes ont été positionnées coté top afin d'être visibles facilement. Certains trous sont nécessaires ici afin de disposer les capteurs comme voulu, ils auront 0.8cm de diamètre afin de pouvoir faire passer un fil. Le plan de masse à été nettement amélioré entre les deux schémas.

Incrustateur :

Avant



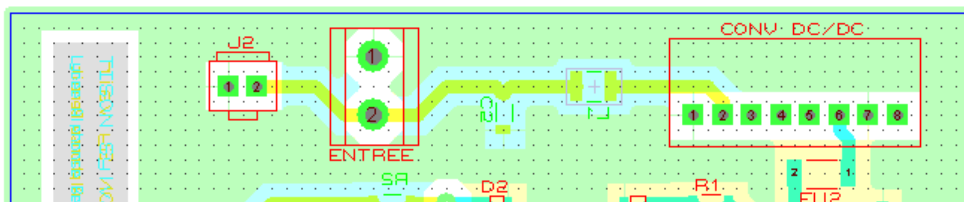
Après



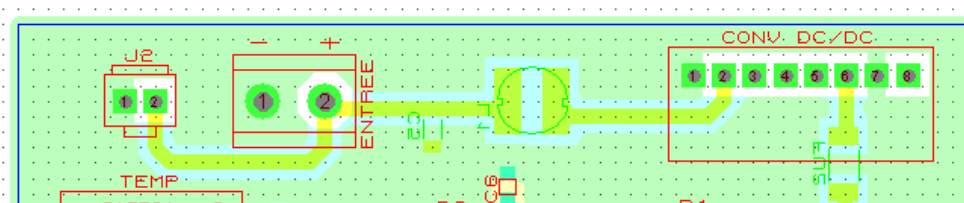
L'incrustateur n'a pas subi de grands changements, il fut seulement déplacé afin de prendre moins d'espace. La piste du MOSI a été modifiée afin de faciliter la soudure de la masse de l'incrustateur lors des phases de test.

Bloc Alimentation :

Avant



Après



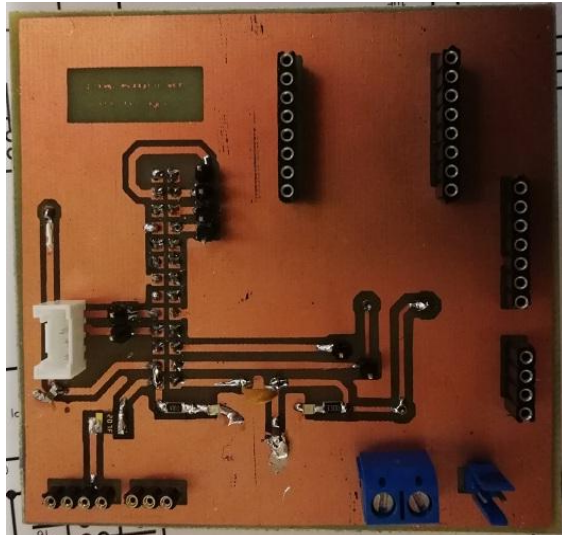
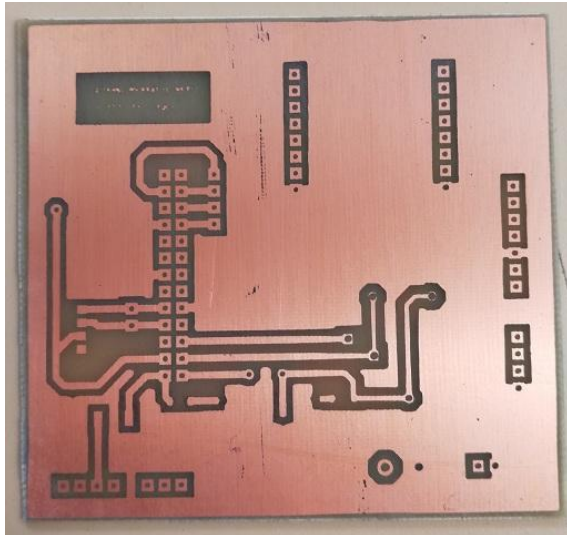
Le connecteur Entrée a été pivoté et (marqué) polarisé afin de faciliter le branchement des batteries. Le fusible FU2 fut déplacé vers la couche bottom afin de faciliter le soudage du convertisseur DC/DC.

Test de la carte d'extension

Photographie des tests visibles en annexe 6 ; Nomenclature des composants en annexe 7

Fabrication :

Après avoir finalisé les schémas de routage des cartes d'extension, nous avons créé des cartes de test au sein de l'établissement afin de connaître certaines erreurs éventuelles avant de commander les cartes finales.



Tests à effectuer en cas de dysfonctionnement : Processus de maintenance curative

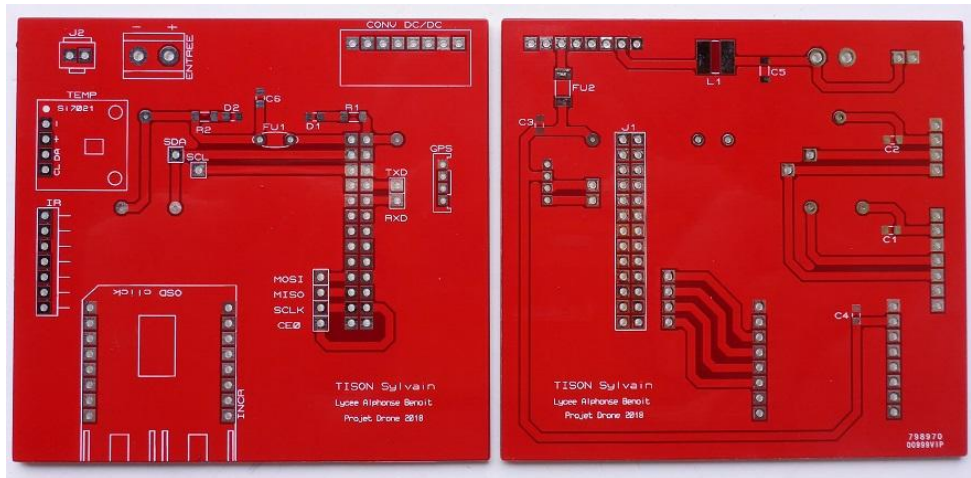
Si les diodes ne s'allument pas et qu'elles ne sont pas défectueuses : Vérifier à l'aide d'un voltmètre que la tension aux bornes de l'alimentation soit de 7V, que la tension à la sortie du convertisseur DC/DC soit de 5V. Si ce n'est pas le cas vérifier les soudures de ces composants (particulièrement les masses). Si le problème persiste tester que le 5V et le 3,3V soient bien transmis dans leurs broches respectives, si ce n'est pas le cas, vérifier qu'il n'y a pas de faux contact entre les pistes et la masse.

Si certains capteurs ne s'allument pas ou ne transmettent pas : Vérifier chaque soudure du capteur et des pins de test afin qu'il n'y ait pas une coupure dans les pistes.

Si la raspberry reboot au démarrage, vérifier qu'elle ne soit pas sous alimentée.

Création du Shield final

Après avoir validé les cartes de test, il a été possible de commander les cartes finales à des professionnels. (https://www.seeedstudio.com/fusion_pcb.html)



A la réception des cartes finales, il a fallu positionner et souder les composants selon un ordre précis :

Les CMS coté Top en premier : Utilisant un four à refusions, il est nécessaire de souder les composants CMS en premier, le côté TOP comportant les plus petits composants j'ai donc commencé par lui. (C6 ; R1 ; R2 ; D1 ; D2)

Les CMS coté Bottom en second : La bobine L1 étant volumineuse j'ai donc choisi de faire cette face après le Top. (L1 ; C1 ; C2 ; C3 ; C4 ; C5 ; FU2)

Les composants traversant Bottom : Ne comportant que J1 et n'étant pas très grand j'ai privilégié cette face.

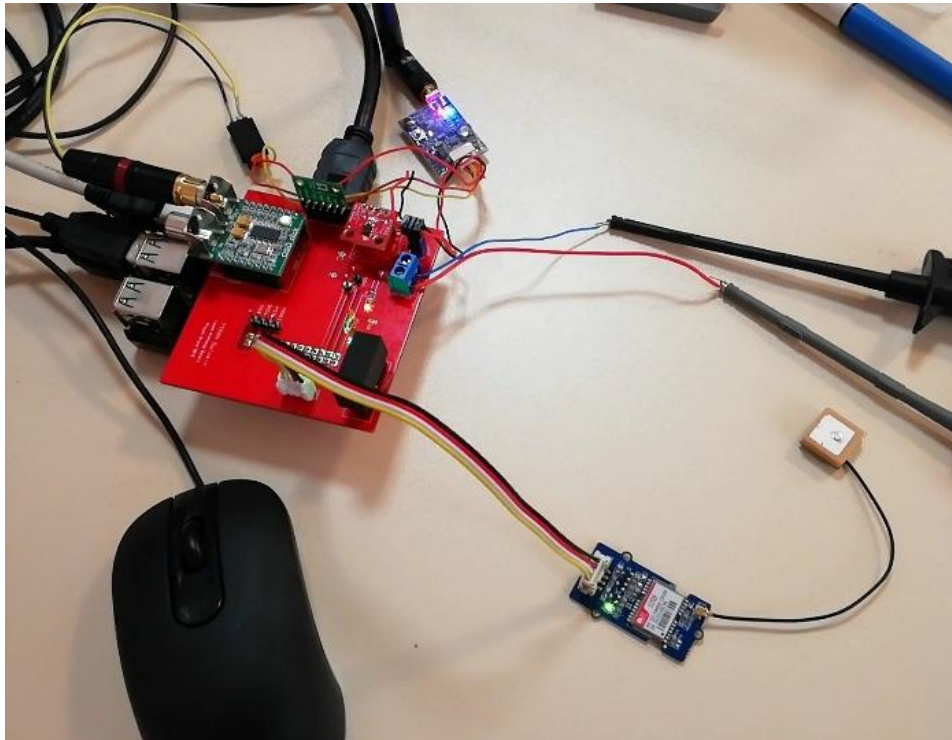
Les composants traversant Top : Comportant beaucoup de composants de tailles relativement élevée, j'ai décidé de finir par ce coté.



Test et mise en place finale :


















Les derniers tests mit en œuvre consistent à alimenter tout les capteurs et la raspberry à l'aide d'une alimentation externe de 7V (correspondant à la tension des batteries) et de 0.7mA.

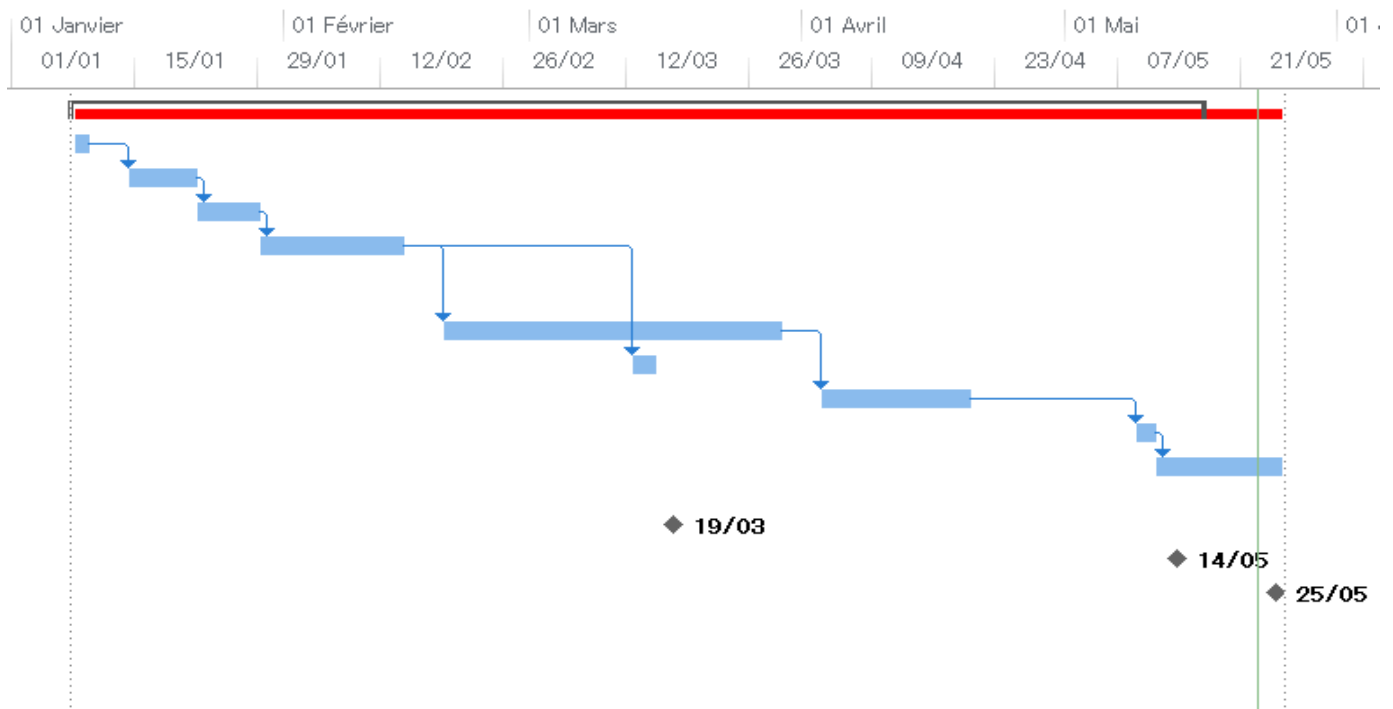
Au cours des semaines suivantes il faudra tester l'alimentation complète avec les batteries du drone.



Annexe 0 : Gant et Journal de bord

Gant Final :

60			Tâches Tison	184 h	Jeu 11/01/18	Mer 16/05/18		TISON Sylvain
61			Recherche et documentation	7 h	Jeu 11/01/18	Ven 12/01/18		
62			Test du Module GPS sur Rpi	17 h	Mer 17/01/18	Mer 24/01/18	61	
63			Codage	13 h	Jeu 25/01/18	Mer 31/01/18	62	
64			Conception du Schéma structurel + Réalisation de la carte d'alimentation	35 h	Jeu 01/02/18	Ven 16/02/18	63	
65			Routage de la carte d'extention	53 h	Mer 21/02/18	Ven 30/03/18	64	
66			Préparation 1ère Revue	13 h	Mer 14/03/18	Ven 16/03/18	64	
67			Test des Circuits Imprimés	33 h	Mer 04/04/18	Ven 20/04/18	65	
68			Préparation 2ème Revue	12 h	Mer 09/05/18	Ven 11/05/18	67	
69			Test et Mise en place carte finale	33 h	Ven 11/05/18	Ven 25/05/18	68	



1. Journal de bord.

<i>Date</i>	<i>Intervenant</i>	<i>Objet : (d) → document drive</i>
• 11/01/2018	• EC1 EC2 EC3 EC4 • IR1 IR2 IR3 • M.Antoine	• Prise de connaissances du Projet Drone - Réunion de groupe (d) - Fiche UART
• 12/01/2018	• EC1 EC2 EC3 EC4	• Prise de connaissances du Projet Drone - Fiche Module Gps Grove 113020003 (d) - Recherche et comparatifs : Gps - Prise de connaissance : SIM28 et Protocole NMEA
• 17/01/2018	• EC3 EC4 IR2	• Tests de la carte GPS - Installation de la carte GPS sur la Raspberry - Initialisation du code C++ - Analyse des trames NMEA (d)
• 18/01/2018	• EC3	• Tests de la carte GPS - Analyse approfondie des trames NMEA (d) - Réussite capture de trame NMEA
• 19/01/2018	• EC3	• Test de la carte GPS - Compte rendu trame GPS (d)
• 24/01/2018	• EC3 EC4 • M.Hortolland	• Test de la carte GPS - Récupération de trame avec Saleae (d) - Étude de solution pour la suite du projet
• 01/02/2018	• EC1 EC2 EC3 EC4 • IR2 IR3 • M.Escuret	• Conception du Shield - Réunion explication du projet avec M.Escuret - Conception Shield Isis (d) - Test des capteurs sur une Raspberry - Récupération de la consommation du drone
• 02/02/2018	• EC3	• Conception du Shield - Préparation du 1 ^{er} compte rendu de projet (d) - Conception Shield Isis (d) - suite

2. Journal de bord.		
<i>Date</i>	<i>Intervenant</i>	<i>Objet : (d) → document drive</i>
• 08/02/2018	• EC1 EC2 EC3 EC4	<ul style="list-style-type: none"> • Conception du Shield - Réalisation schéma Fritzing (d) - Conception broches GPS Isis (d) - suite
• 09/02/2018	• EC3	<ul style="list-style-type: none"> • Conception du Shield - Conception Shield Isis (d) - Fin - Préparation à la 1^{ère} revue de projet
• 15/02/2018	• EC1 EC2 EC3 EC4	<ul style="list-style-type: none"> • Conception du Shield - Prise de connaissance du Drone - Etude de la répartition des composants sur le Drone
• 21/02/2018	• EC3	<ul style="list-style-type: none"> • Conception du Shield - Conception du Shield Ares (d) - Début - Finalisation de la 1^{ère} Revue de Projet
• 28/02/2018	• EC1 EC2 EC3 EC4	<ul style="list-style-type: none"> • Conception du Shield - Conception de Shield Ares (d) - Suite - Scan plaque drone (d)
• 30/03/2018	• EC1 EC2 EC3 EC4	<ul style="list-style-type: none"> • Conception du support Rpi - Mesures du support (d) - Conception du shield Ares (d) - Fin
• 04/04/2018	• EC3	<ul style="list-style-type: none"> • Conception du shield - Création du 1^{er} shield de test
• 05/04/2018	• EC3	<ul style="list-style-type: none"> • Conception du shield - Nomenclature (d) - Premiers test sur le shield
• 11/04/2018	• EC3	<ul style="list-style-type: none"> • Conception du Shield - Perçage / Soudage de la carte shield personnelle
• 18/04/2018	• EC3	<ul style="list-style-type: none"> • Conception du Shield - Test shield personnel – Fin (d) Vidéo

Annexe 1 : Code du module GPS



```
// Récupération des données d'un module GPS par Rpi
#include <iostream>
#include <string.h>
#include <errno.h>
#include <wiringPi.h>
#include <wiringSerial.h>

using namespace std;

// g++ -o GPS3 -lwiringPi GPS3.cpp

int main ()
{
    while(1) {
        int fd;

        if (wiringPiSetup () == -1)
        {
            cout<<"Impossible de démarrer wiringPi. Erreur : "<<strerror (errno);
            return 1 ;
        }

        if ((fd = serialOpen ("/dev/ttyAMA0", 9600)) < 0)
        {
            cout<<"Impossible d'ouvrir le port /dev/ttyAMA0. Erreur : "<<strerror (errno);
            return 1 ;
        }

        char data;

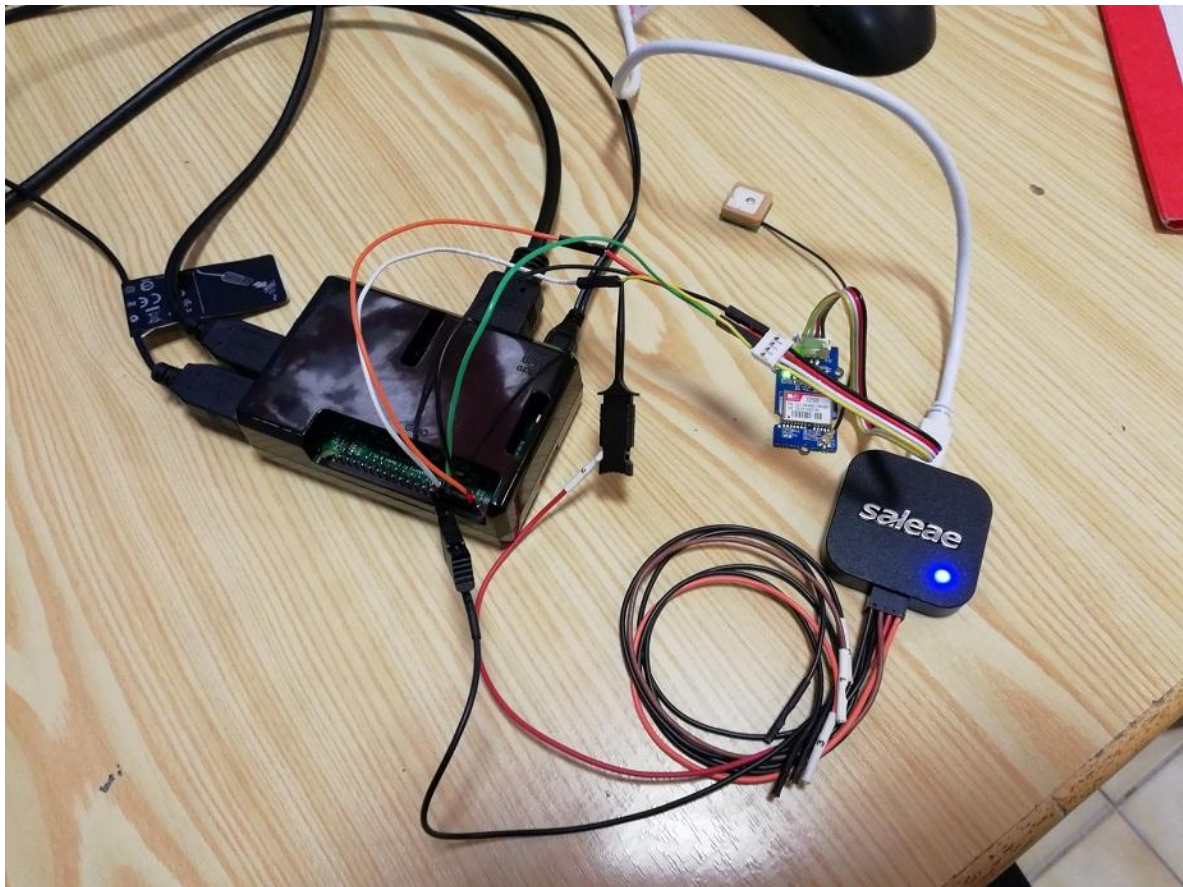
        for (int i(0); i<72; i++)
        {
            data = serialGetchar(fd);    // Lecture et stockage dans data
            cout<<data<<" ";
        }
        cout<<endl;

        serialFlush(fd);    // Effacement buffers série avant lecture Rx

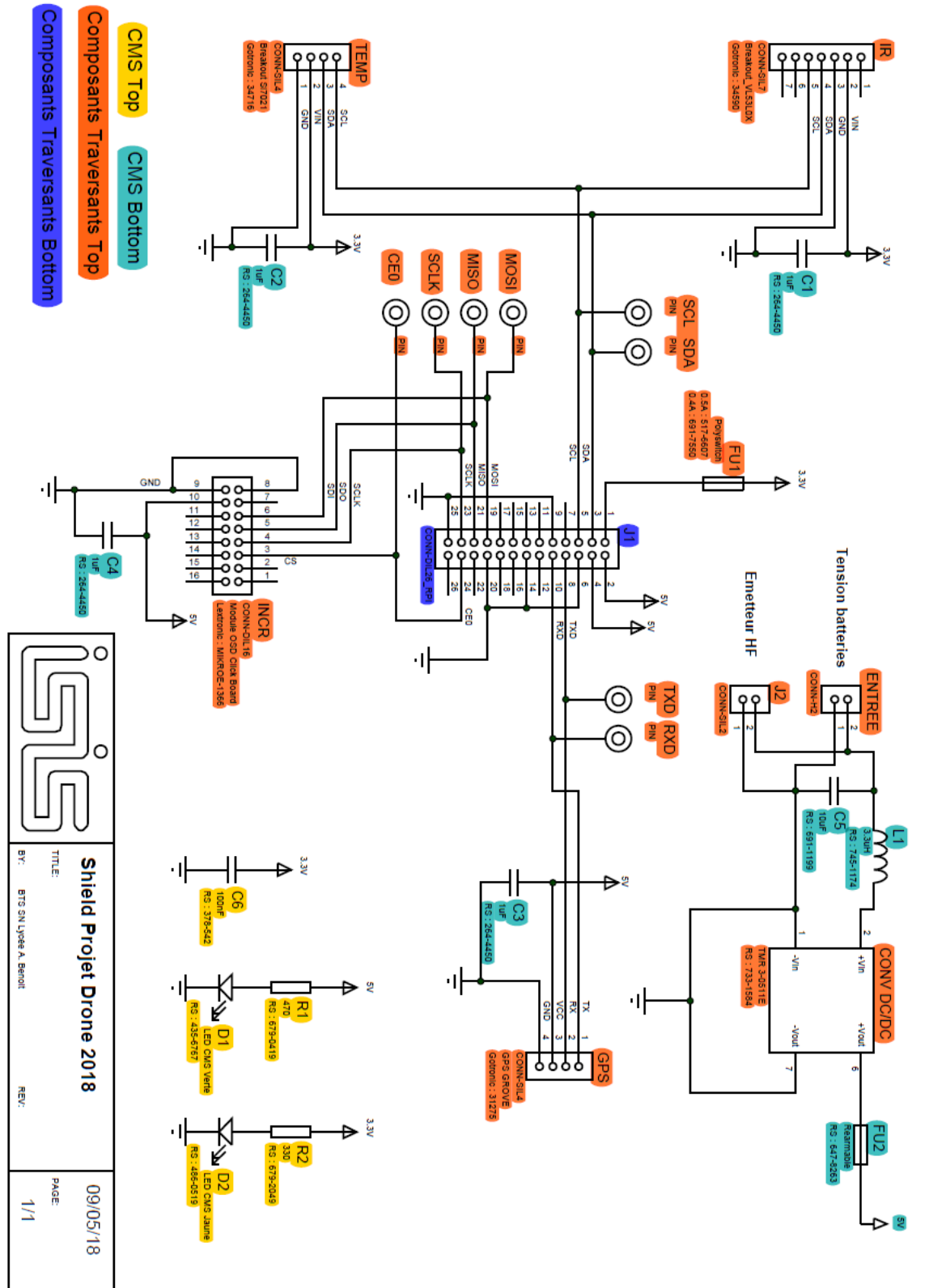
        serialClose(fd);    // Fermeture de l'interface série
    }
    return 0 ;
}
```

Enregistrement du fichier « /home/pi/Projet/Codes/GPS/GPS3.cpp »... C++ Largeur des tabulations : 8 Lig 35, Col 52 INS

Annexe 2 : Capture de la trame NMEA avec l'analyseur logique Saleae :

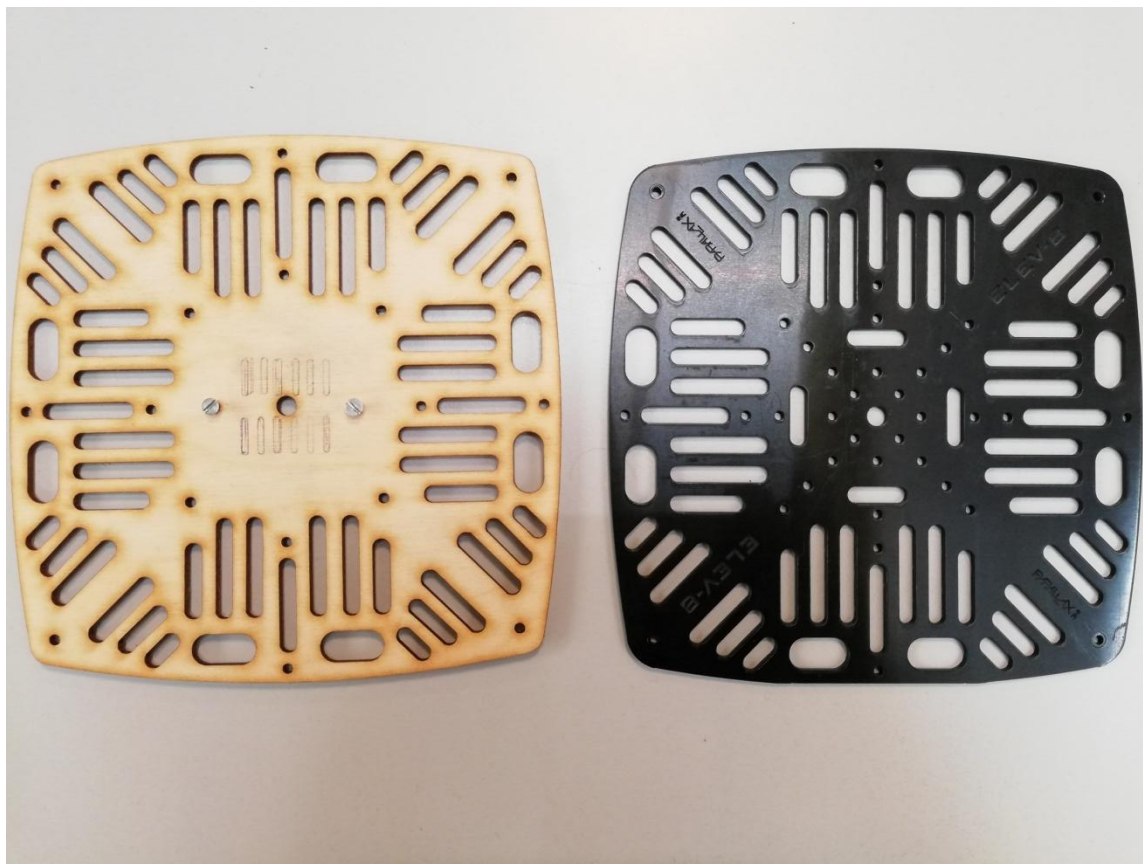


Annexe 3 : Schéma Structurel

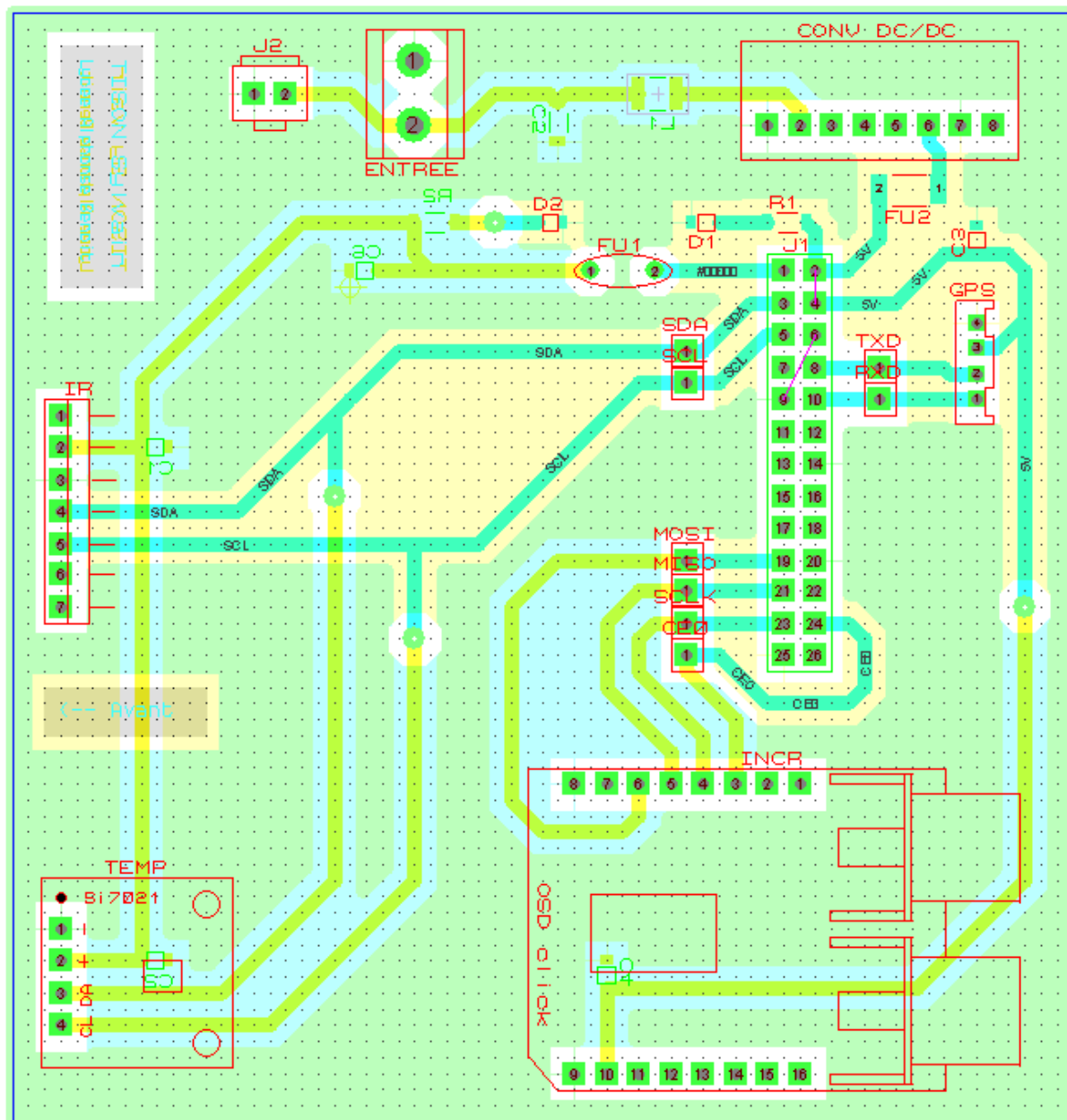


	Shield Project Drone 2018 BY: BTS SN Lycée A. Benoit REV:	09/05/18 PAGE: 1/1
---	--	-----------------------

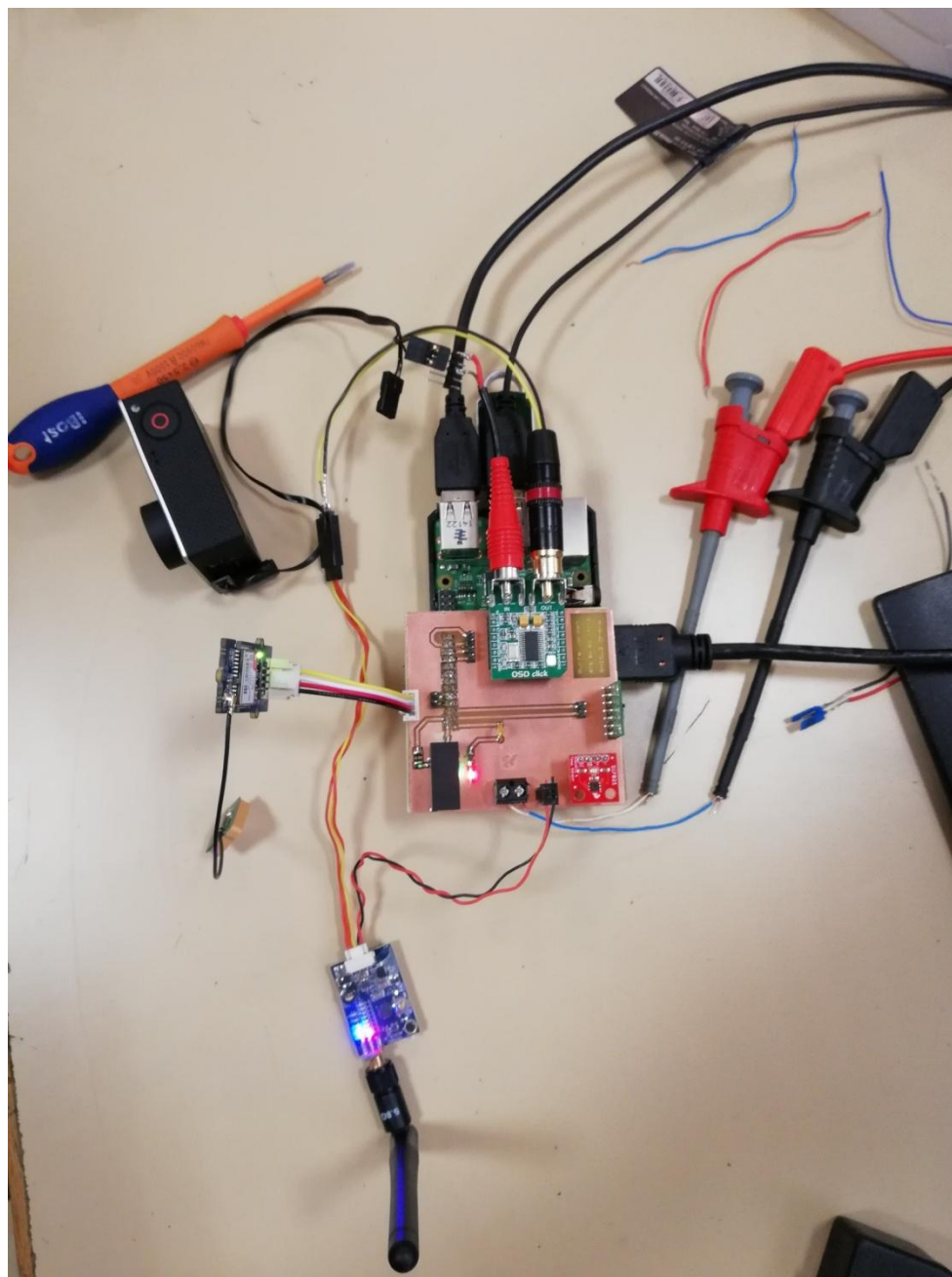
Annexe 4 : Support du drone



Annexe 5 : Premier schéma de routage : 28/03



Annexe 6 : Essai du PCB de test avec tout les capteurs



Annexe 7 : Nomenclature

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		435-5767	0,132	0,132
D2	LED CMS Jaune	HSMY-C170			Radiospares	2012		1		488-0519	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		647-5283	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		800-7748	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 flick 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	Connecteur 2x13	3,54mm			Gotronic			1		8017	0,580	0,580
J2	Embrase 2 contacts	2,54mm			Gotronic			1		681-1871	0,441	0,441
ENTREE	Bornier 2 contacts	3,5mm			Radiospares			1		710-0444	0,520	0,520
Port Grove	Embrase CI 4 contact	2mm			Gotronic			1		31232	0,130	0,130
TOTAL HT FEUILLE												101,612
TOTAL HT GLOBAL												101,612
											COEFFICIENT	0,000

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

PROJET DRONE : BOTTIGLIERO
NICOLAS : BTS SYSTÈMES
NUMÉRIQUES



II – EC:MESURER (DISTANCE)

Introduction :

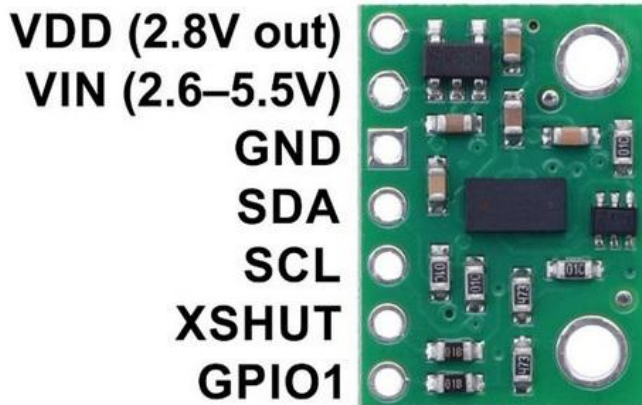
Dans ce projet je suis l'EC 4 dans l'équipe 2, je m'occupe de la gestion du capteur de distance infrarouge.

Ce capteur aura pour utilité de mesurer la distance entre le drone et l'obstacle devant lui.

Étudiant	Liste des fonctions assurées par l'étudiant	Installation :
EC4 IR	<p>EDD : UC Mesurer (la distance)</p> <p>Etudier la solution 2017.</p> <p>Valider une nouvelle solution.</p> <p>Concevoir/Réaliser/Tester une nouvelle carte d'extension pour Rasperry Pi intégrant tous les breakout des capteurs et autres.</p> <p>Développer une application permettant de vérifier, le bon fonctionnement de la mesure de distance.</p>	<p>- Rpi : bus I2C et librairie C/C++ BCM2835.</p> <p>Mise en œuvre :</p> <p>- Valider par prototypage rapide le capteur de distance proposé.</p> <p>Réalisation :</p> <p>- Participer avec les autres étudiants EC à la conception d'un schéma commun intégrant tous les composants de l'EDD.</p> <p>- Concevoir une carte électronique personnelle compatible avec tous les contrats EC.</p> <p>Documentation :</p> <p>- Bibliothèque C/C++ d'accès au capteur de distance.</p> <p>- Documents de fabrication de la carte. Ces documents devront permettre une fabrication industrielle du circuit imprimé.</p>

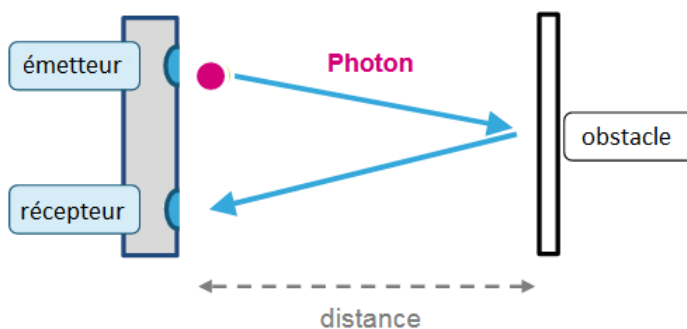
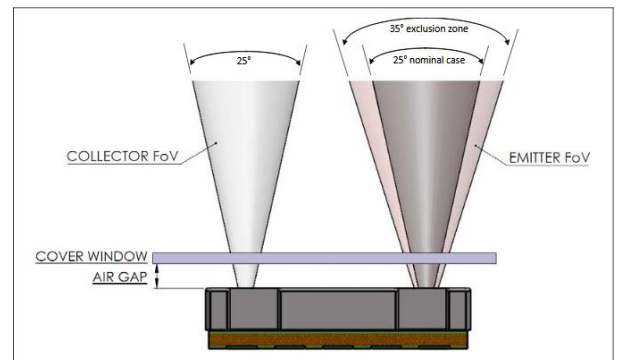
Pour ce projet, le capteur de distance à ultrason utilisé par Grimoud Yohan l'année précédente a été remplacé par un capteur de distance infrarouge pour des raisons d'encombrement, le capteur à ultrason était beaucoup plus gros que le capteur à infrarouge et son coût était également plus élevé.

Capteur de distance infrarouge VL53L0X :



Le VL53L0X est un capteur permettant la mesure de distance grâce à de l'infrarouge.

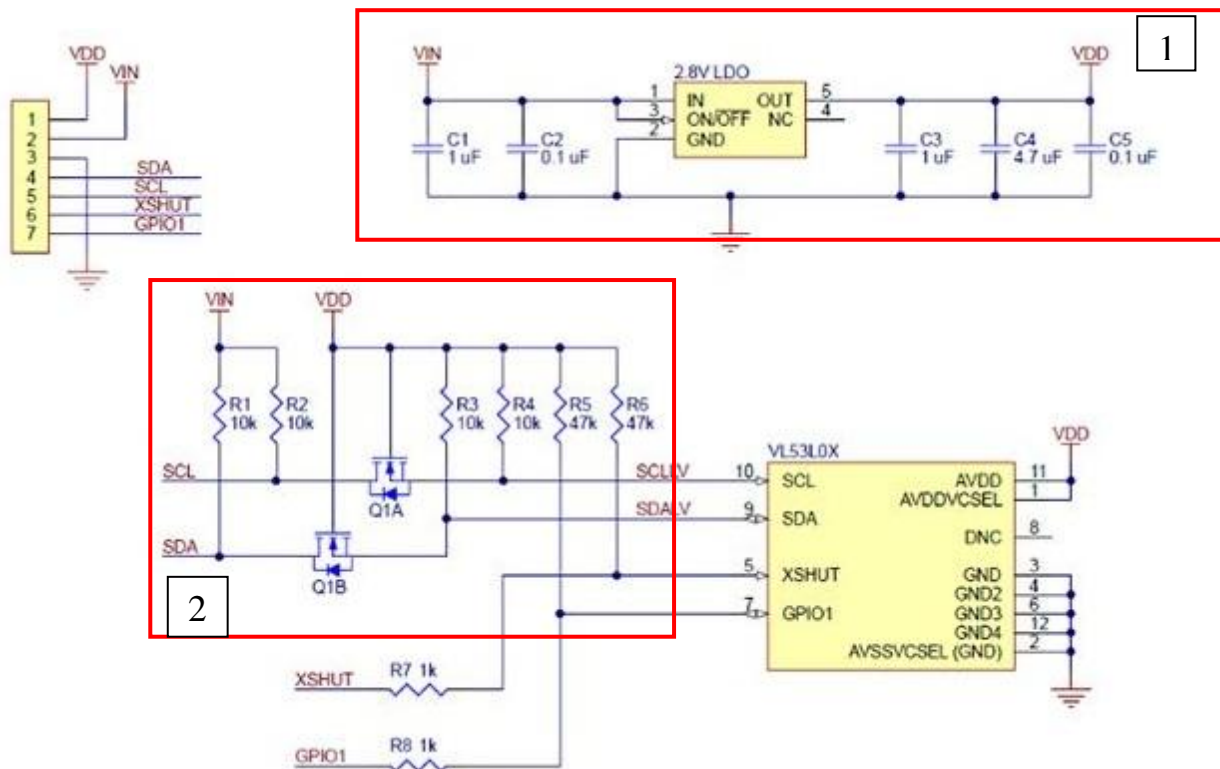
Le capteur est constitué de deux lentilles une (à droite) qui permet l'émission du faisceau Infrarouge et l'autre (à gauche) permettant la réception du signal



L'émetteur envoie un signal infrarouge qui rebondit sur l'obstacle et revient au récepteur. Il calcule ensuite la distance.

La VL53L0X communique en I²C, son adresse par défaut est 0x29 sur 7 bits, elle est changeable grâce au programme.

Schéma structurelle du VL53L0X :

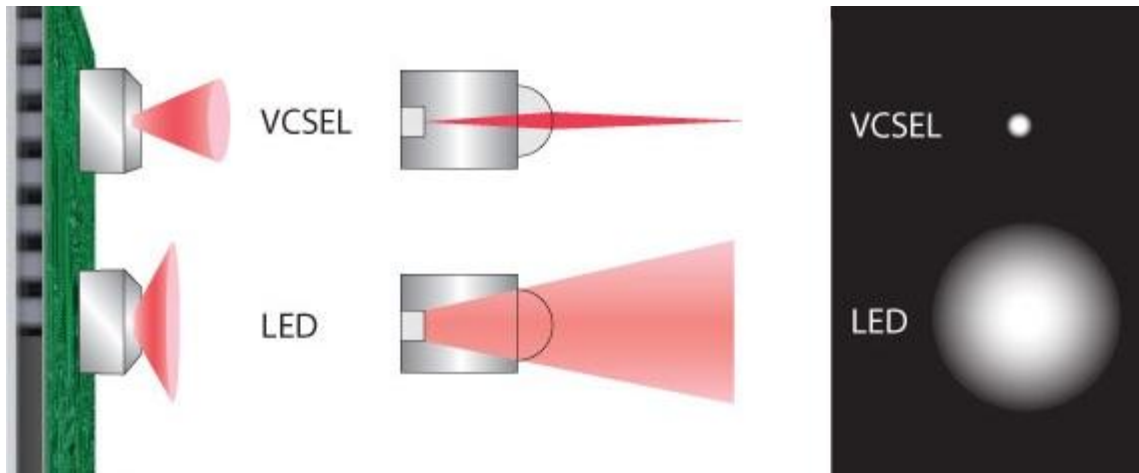


Le VL53L0X a sur son breakout, son propre adaptateur de tension (1), qui convertit les 3,3 V reçue en 2.8 V qui est nécessaire pour alimenter le capteur.

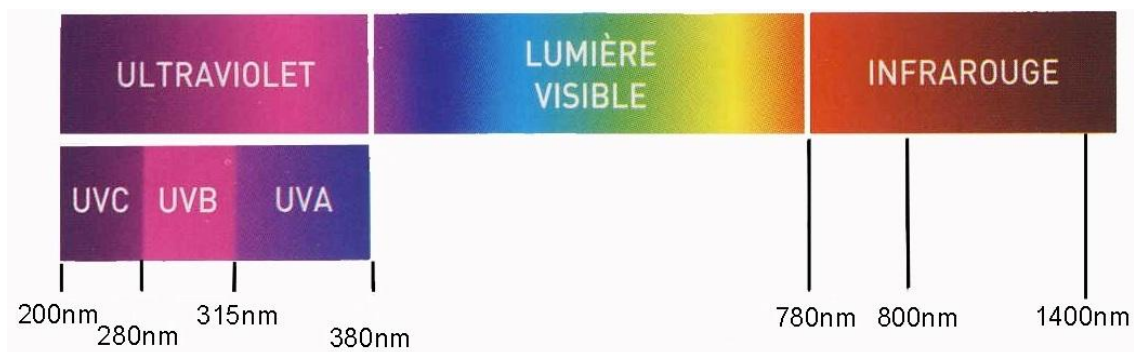
Un level shifter y est également inclus (2) pour l'I2C .

Infrarouge :

Le VL53L0X est un capteur utilisant l'infrarouge et est équipé d'une diode laser à cavité verticale émettant par la surface, ou VCSEL, qui a la particularité d'émettre par la tranche contrairement à une led Classique.



Elle a une longueur d'onde de 940nm, ce qui n'est pas visible à l'œil nu.



Une photo prise avec mon téléphone dont l'appareil photo permet de voir l'infrarouge.

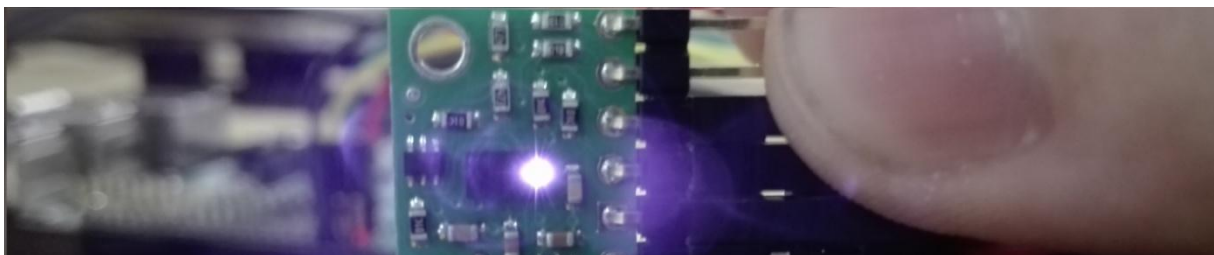
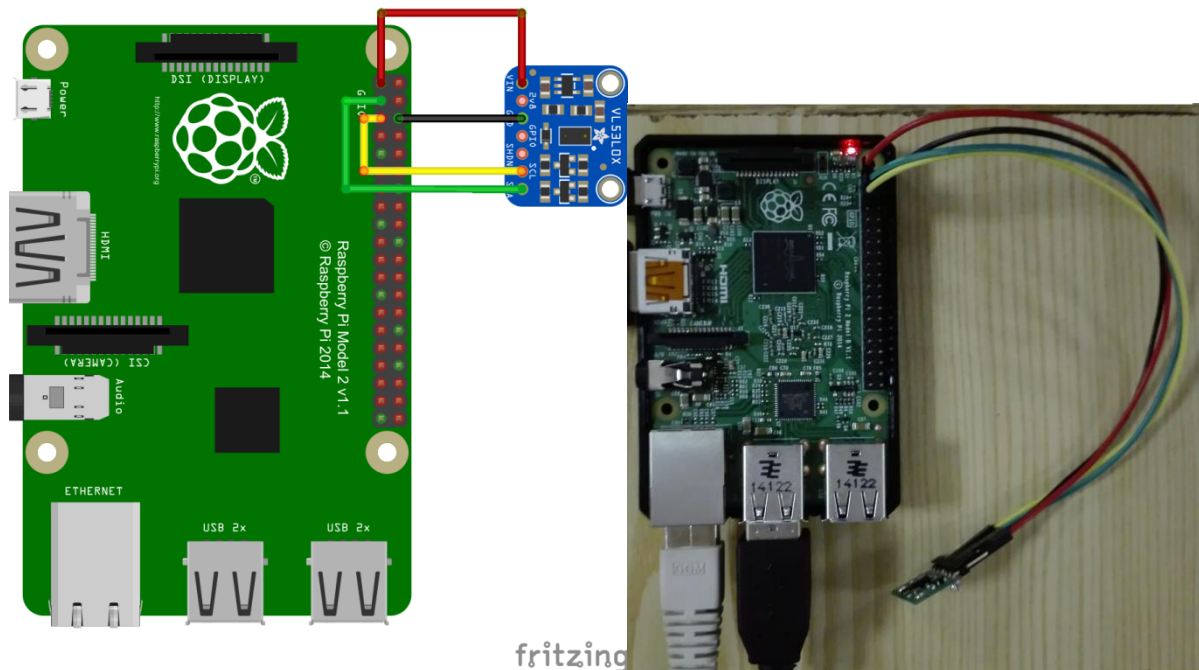


Schéma de câblage rapide :

Le symbole du schéma Fritzing n'est pas le même que celui utilisé sur le projet Drone mais les connectiques sont équivalentes.

Schéma Fritzing :



Connections :

Raspberry Pi	VL53L0X
3.3 V Power (Pin 1)	VIN
SDA1 (Pin 3)	SDA
SCL1 (Pin 5)	SCL
Ground (Pin 9)	GND

Mise en œuvre de l'exemple avec Cmake sur Rpi :

Cmake est nécessaire pour compiler le programme d'exemple.

L'exemple utilisé peut-être obtenu à cette adresse :

<https://github.com/mjbogusz/vl53l0x-linux>

Pour installer l'exemple utilisé pour le test du capteur il faut passer par le logiciel Cmake.

Il faut d'abord installer Cmake pour cela, sous Raspbian ouvrir une console et taper la commande suivante :

```
pi@raspberrypi:~ $ sudo apt-get install cmake
```

L'installation se lance alors, une fois terminé on pourra installer la librairie.

Télécharger l'exemple sur Github et décompresser dans un sous-répertoire sur la Rpi, déplacez-vous dedans.

```
pi@raspberrypi:~ $ cd /home/pi/Travail/Drone/vl53l0x-linux-master
```

Une fois dans le bon répertoire entré la commande suivante pour compiler les programmes d'exemples

```
pi@raspberrypi:~ $ cd build
```

```
pi@raspberrypi:~ $ cmake ..
```

```
pi@raspberrypi:~ $ make
```

Une fois les commandes rentrées plusieurs exemples sont utilisables ils ont décrits sur le Github, ici nous allons utiliser l'exemple « singleMinimal ».

```
pi@raspberrypi:~/Travail/Drone/vl53l0x/build $ ./examples/singleMinimal
```

La mesure commence alors, les distances mesurées sont en millimètres

```
pi@raspberrypi:~/Travail/Drone/vl53l0x/build $ ./examples/singleMinimal
53
56
54
52
56
53
53
53
57
```


Programme de test :

Programme de base :

```
#include <iostream>
#include "VL53L0X.hpp"

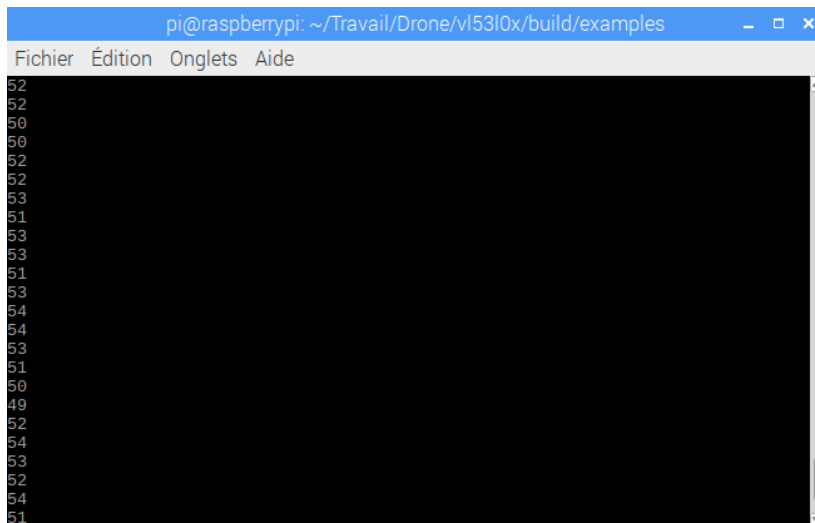
int main() {
    VL53L0X sensor;
    sensor.init();
    sensor.setTimeout(200);

    for (int i = 0; i < 1000; ++i) {
        uint16_t distance = sensor.readRangeSingleMillimeters();
        if (sensor.timeoutOccurred()) {
            std::cout << "timeout!" << std::endl;
        } else {
            std::cout << distance << std::endl;
        }
    }

    return 0;
}
```

Ce programme en C++ était fourni par le créateur de l'exemple, il m'a permis de tester le capteur et de voir s'il affiche des mesures cohérentes.

Le programme n'affiche les mesures qu'en millimètres et utilise une librairie spécialement créée pour lui, la VL53L0X.hpp.



Nous constatons sur la capture d'écran une mesure simple en mm, ici 54 mm , 5,4 cm.

Programme modifié :

```
#include <iostream>
#include "VL53L0X.hpp"
#include <unistd.h>

int main() {
    VL53L0X sensor;
    sensor.init();
    sensor.setTimeout(200);

    while(1) {
        uint16_t distance = sensor.readRangeSingleMillimeters();
        if (sensor.timeoutOccurred()) {
            std::cout << "timeout!" << std::endl;
        } else {
            usleep(500000); // 0,5 sec
            if (distance < 1000) {
                std::cout << (distance * 0.1) << " cm" << std::endl; //Distance en cm
            } else {
                std::cout << (distance * 0.001) << " m" << std::endl; //Distance en m
            }
        }
    }

    return 0;
}
```

J'ai modifié le programme pour qu'il affiche les mesures en centimètre puis à une certaine distance, il affiche la mesure en mètres, j'ai également ajouté une temporisation de 0,5 seconde par mesure.

```
pi@raspberrypi: ~/Travail/Drone/VL53L0X_V5
Fichier  Édition  Onglets  Aide
7.4 cm
7.2 cm
7.3 cm
14.4 cm
34.3 cm
74.3 cm
58.8 cm
45 cm
41 cm
41.4 cm
40.4 cm
43.7 cm
46.4 cm
48.5 cm
78.9 cm
1.003 m
1.092 m
1.056 m
1.093 m
95.1 cm
68.5 cm
55 cm
34.8 cm
18.3 cm
```

Sur cette capture on peut constater l'affichage en mètres de la mesure au-dessus de 100 centimètres.

Problème rencontré

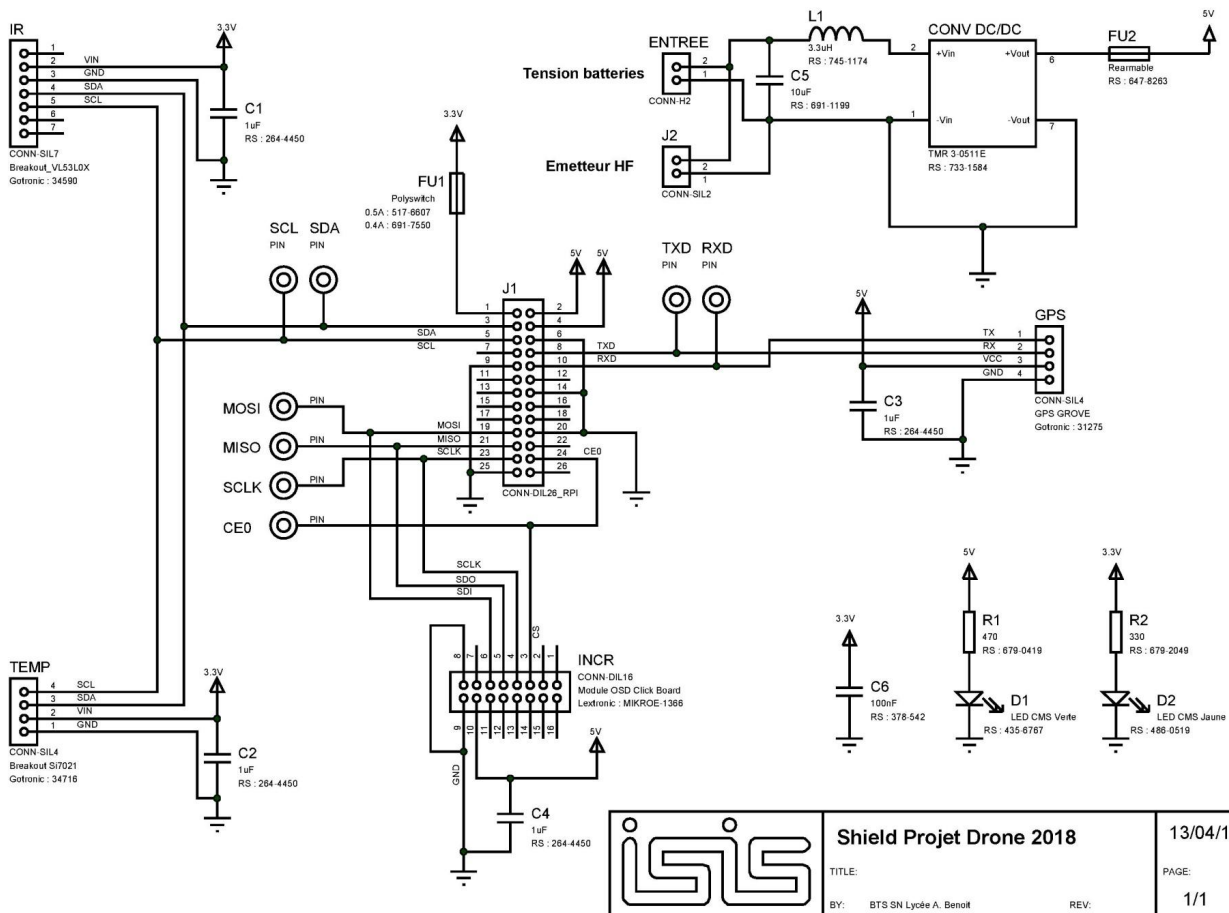
Lecture de trame

ST Microelectronic ne donne pas de documentation assez détaillée pour pouvoir lire correctement les trames envoyées ou reçues par le VL53L0X.

J'ai pu comprendre grâce à l'aide de mes professeurs que les données lues dans la trame étaient envoyées directement par le programme mais sans trop savoir ce que cela voulait dire.

Conceptions du circuit imprimé :

Schéma structurel :



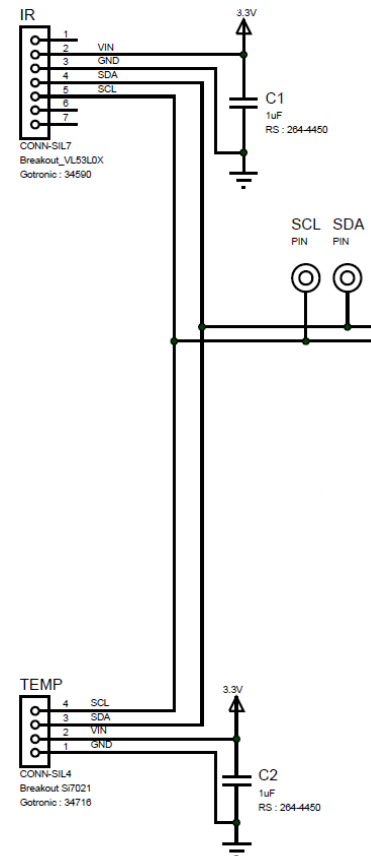
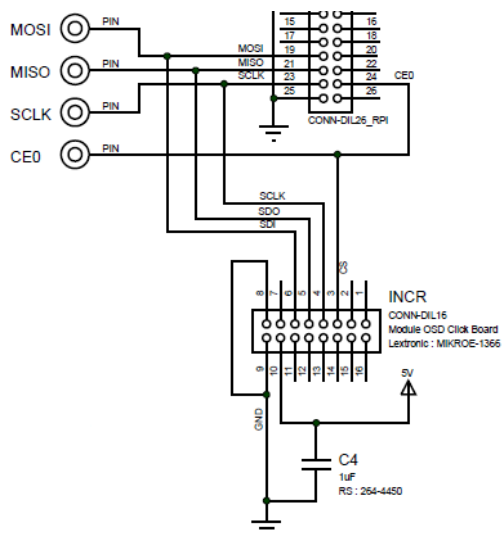
A partir de la BDD donnée dans le contrat (Annexe 4), avec le groupe EC du projet drone nous avons en commun, créé un schéma structurel dont je vais détailler les points importants.

Les capteurs :

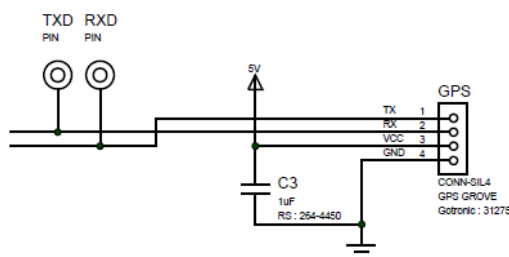
La partie I2C comporte le capteur de distance infrarouge et le capteur de température/humidité.

Grâce à l'I2C qui nous donne le choix de leurs adresses on peut les placer sur les mêmes broches de la raspberry, 0X29 pour le capteur a infrarouge et 0X40 pour le capteur de température.

Les condensateurs sont là pour aider à l'alimentation des deux capteurs, des pins de test commune on était également mise sur les broches SDA et SCL des deux capteurs.

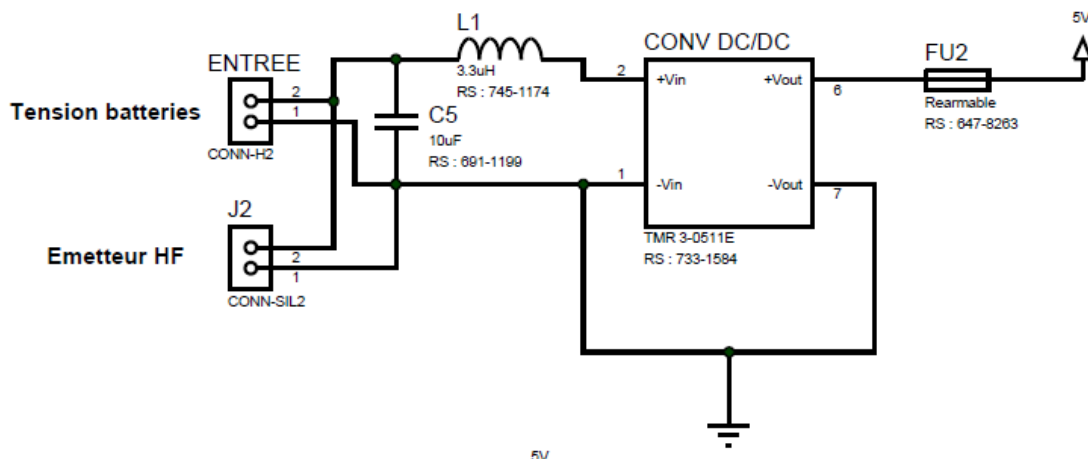


L'incrustateur est en SPI, quatre pins teste on était mis à chaque sortie et un on y a également mis un condensateur.



Le module GPS fonctionne en UART, encore une fois un condensateur y est mis pour aider à l'alimentation.

Convertisseur DC/DC :



Comme nous allons alimenter directement la raspberry par un des ports 5V, nous avons choisi un convertisseur DC/DC, protégé par un fusible.

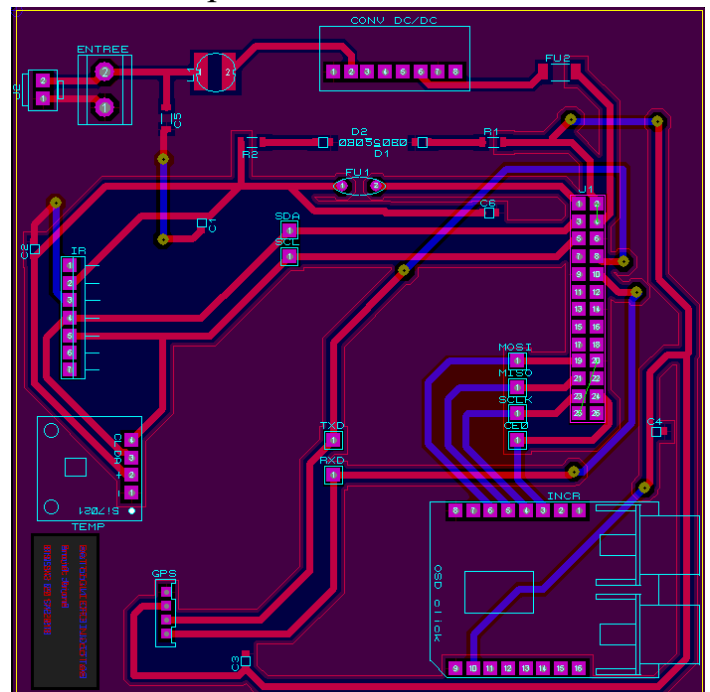
Routage sous ARES :

D'après le schéma ISIS fait précédemment que l'on peut retrouver dans l'annexe 2, j'ai pu commencer le routage sous ARES

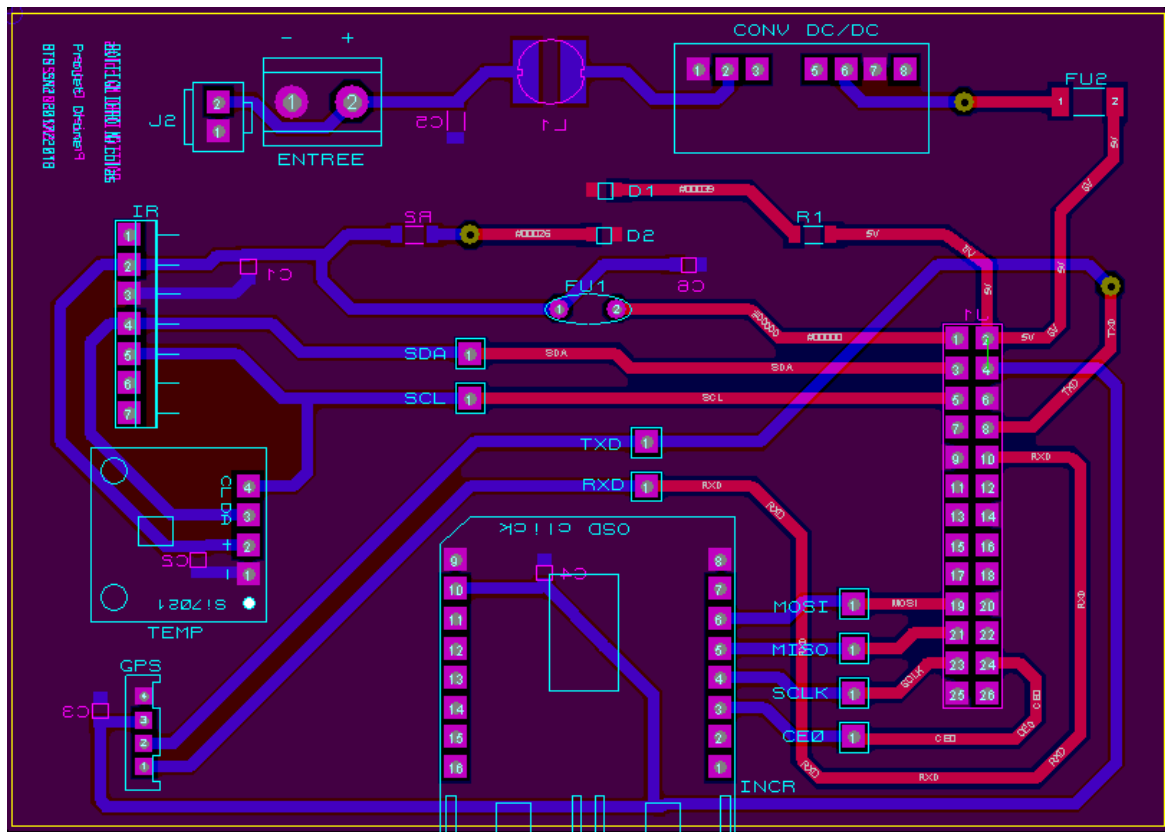
Pour la carte test j'étais partie sur une carte de 10 cm par 10 cm.

Cette carte fonctionnait très bien, mais plusieurs soucis comme sa taille, pose des problèmes, la partie du cotée de l'incrustateur touche les ports USB de la raspberry et crée des court circuits.

Des pistes étaient coupée, car trop près du bord.



Après plusieurs tests concluant sur la carte test je l'ai corrigé pour en arriver à ça :

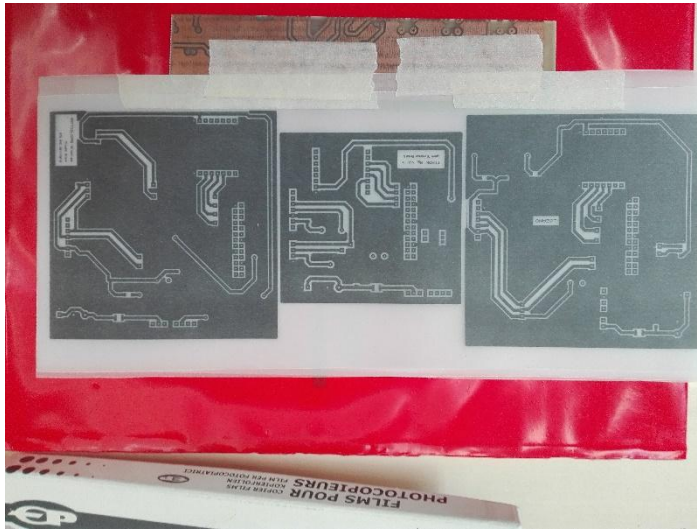


La carte a été rétrécie pour que les ports USB de la raspberry ne touche plus le plan de masse. Les pistes sortantes des composants et de la raspberry ont été redressées.

Tous les condensateurs ont été placés à côté des composants afin de stabiliser le courant qui les alimente, les mettre trop loin ne servirait à rien.

Le capteur infrarouge a été placé à l'avant de la carte pour des raisons évidentes, mesurer les obstacles à l'avant du drone.

Etapes de conception de la carte test :



Les schémas ARES réalisés, Le professeur les imprime sur du papier-calque avec une imprimante standard. On pose ensuite les calques sur une plaque recouverte de cuivre et d'une couche de produit chimique sensible aux UV. Le circuit est ensuite placé dans une insoleuse, qui va venir grâce à l'UV fragiliser la plaque aux endroits exposés.

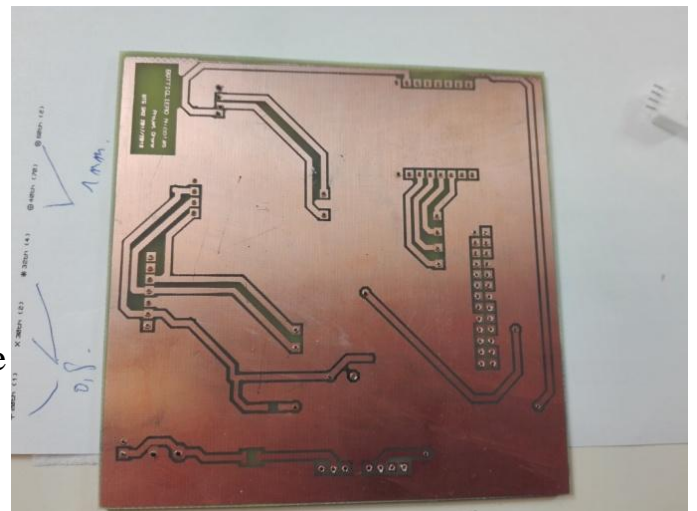
Comme pour la photographie, après être passé aux UV, le circuit va être passé dans un liquide "révélateur", qui va petit à petit révéler les pistes du circuit imprimé.

La dernière étape de la transformation nécessite l'utilisation de perchlorure de fer pour faire dissoudre les parties de la plaque qui étaient exposées aux UV.

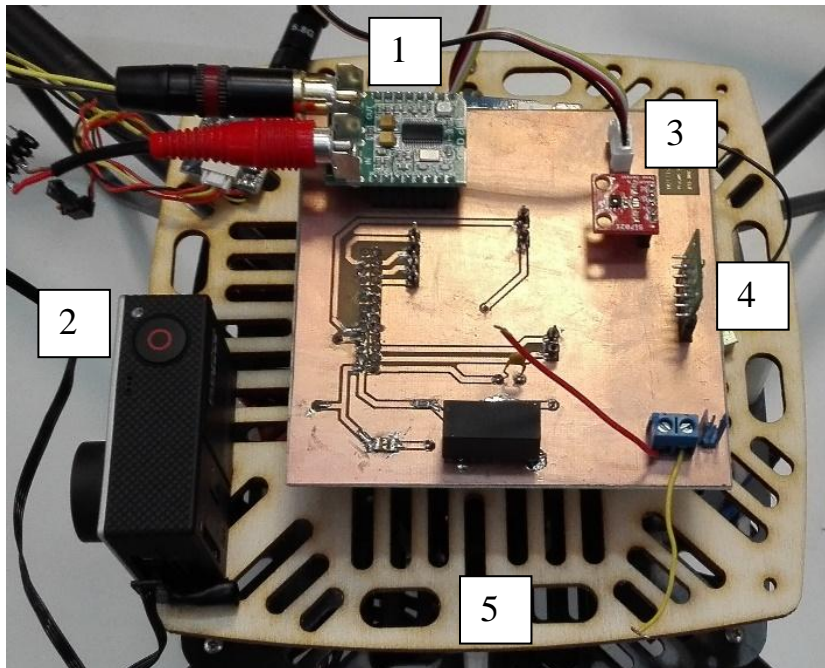
Cette étape passée il ne nous reste plus qu'à frotter le circuit imprimé avec de l'alcool à 90° pour le nettoyer, il faut également limer les bords.

La carte nettoyée, on va venir percer là où les composants traversant vont être placés.

Ici des trous de 0.8 mm et de 1 mm on était nécessaire.



Après le perçage on viendra souder tous les composants et réunir les différents capteurs ainsi que l'incrustateur vidéo.



Ci contre la carte terminée disposé sur le drone.

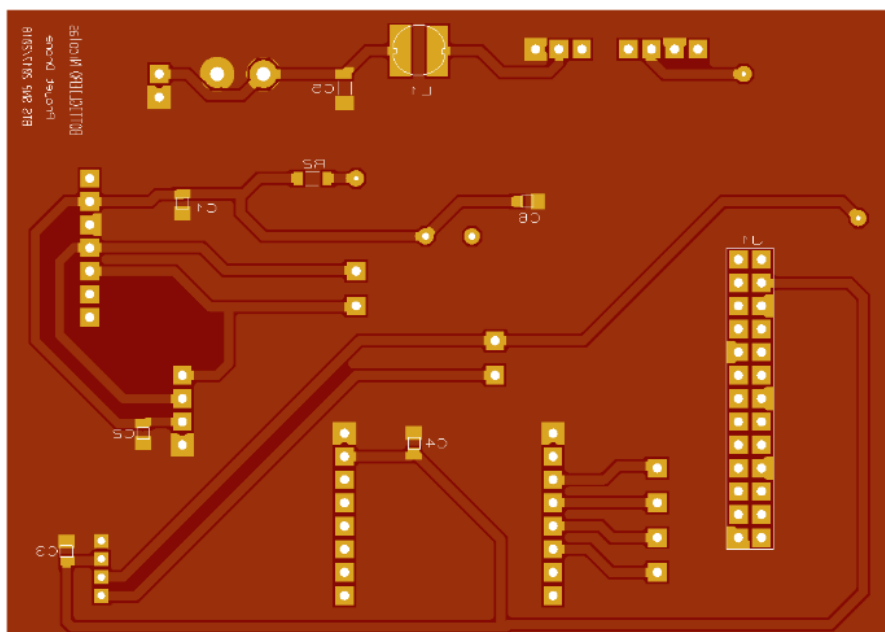
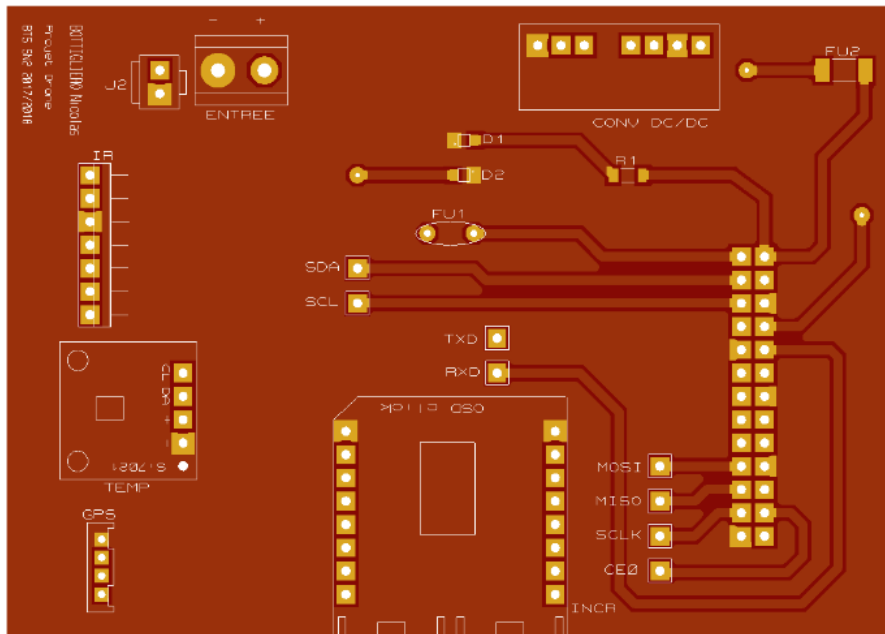
On peut voir sur la photo, différents éléments posés sur le Drone :

- 1 : L'incrustateur vidéo
- 2 : La camera Gopro
- 3 : Le capteur thermique
- 4 : Le capteur de distance infrarouge
- 5 : Le convertisseur DC/DC //

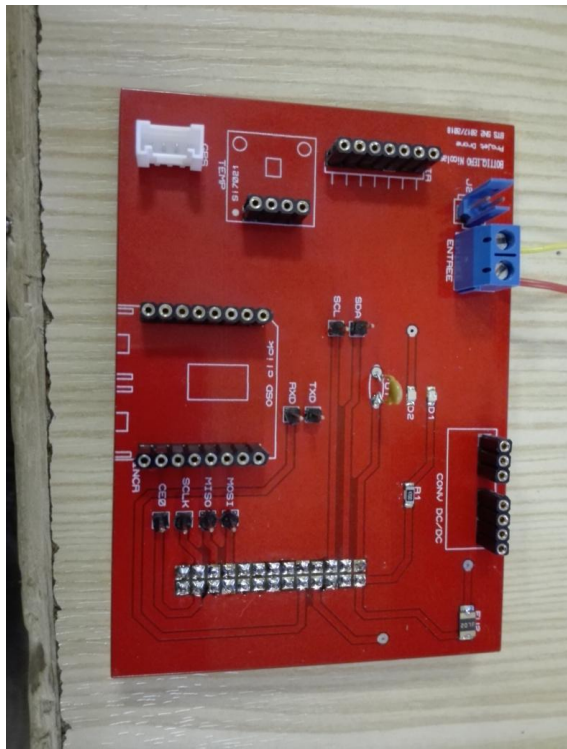
On peut vérifier également si un routage peut fonctionner en exportant un fichier Gerber de notre routage depuis ARES et les visionner sur un site qui nous propose un « Gerber Viewer », tel que <http://www.gerber-viewer.com/>.

Mais pour ma part ce site ne marchait pas pour des raisons inconnues j'ai donc dû aller sur www.jlcpb.com et voici le résultat.

On peut y voir les deux cotés de la carte, TOP et Bottom .

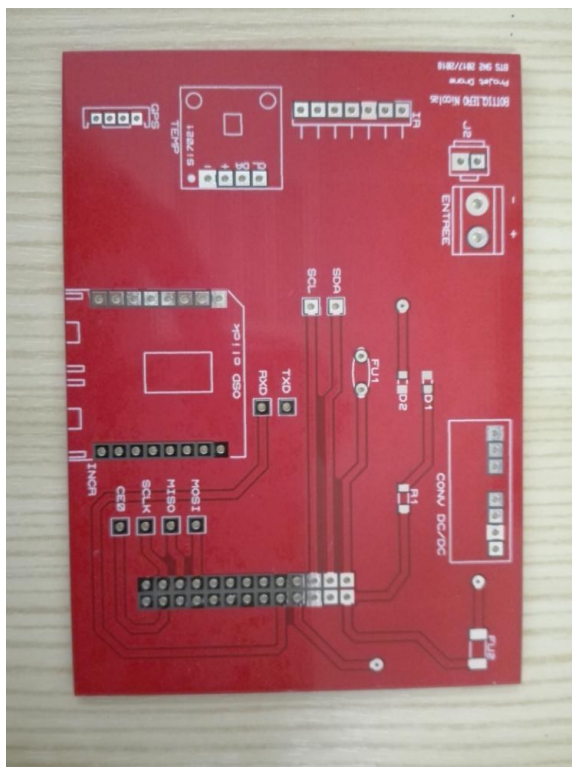


Voici la carte reçue



Et avec les composants mis en place.

Dans l'annexe (5) les étapes pour souder les composants sont détaillées.



Analyse de trame :

Pour la partie physique j'ai récupéré avec l'analyseur Logique Saleae la partie de la trame ou on pouvait lire la mesure de distance du capteur :

Mesure	Saleae	calcule
145mm	Setup Read to [41] + ACK 0 + ACK 145 + NAK	145+0=145
311mm	Setup Read to [41] + ACK 1 + ACK 55 + NAK	55+255=310
404mm	Setup Read to [41] + ACK 1 + ACK 148 + NAK	148+255=403
489mm	Setup Read to [41] + ACK 1 + ACK 233 + NAK	233+255=489

On peut remarquer que quand la mesure excède les 250 mm le premier octet de la trame se met à 1.

La valeur ne peut s'écrire que sur 8bit et en configurant cet octet on ajoute 8bits supplémentaires.

La dernière partie du tableau met en évidence le calcule effectué pour retrouver les bonnes valeurs.

Création de la plaque :

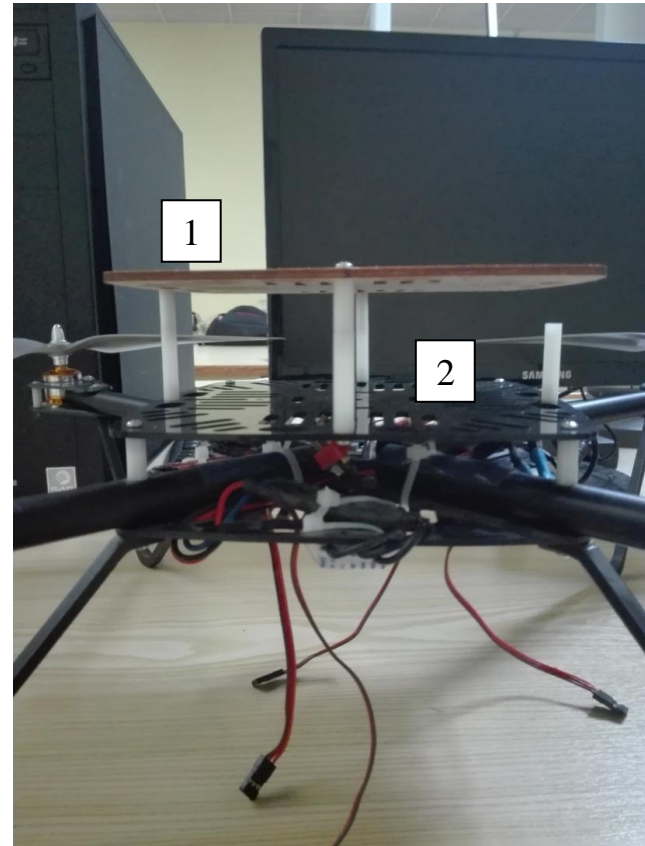
Pour la mise en place de la raspberry et de notre circuit imprimé sur le drone l'équipe de projet EC a réfléchi ensemble pour trouver une solution, on en a conclu qu'il faudrait rajouter un étage sur le drone pour permettre de poser la raspberry équipé du circuit et ses capteurs.

Sur le premier étage viendra se poser la Rpi avec ses capteurs(1).

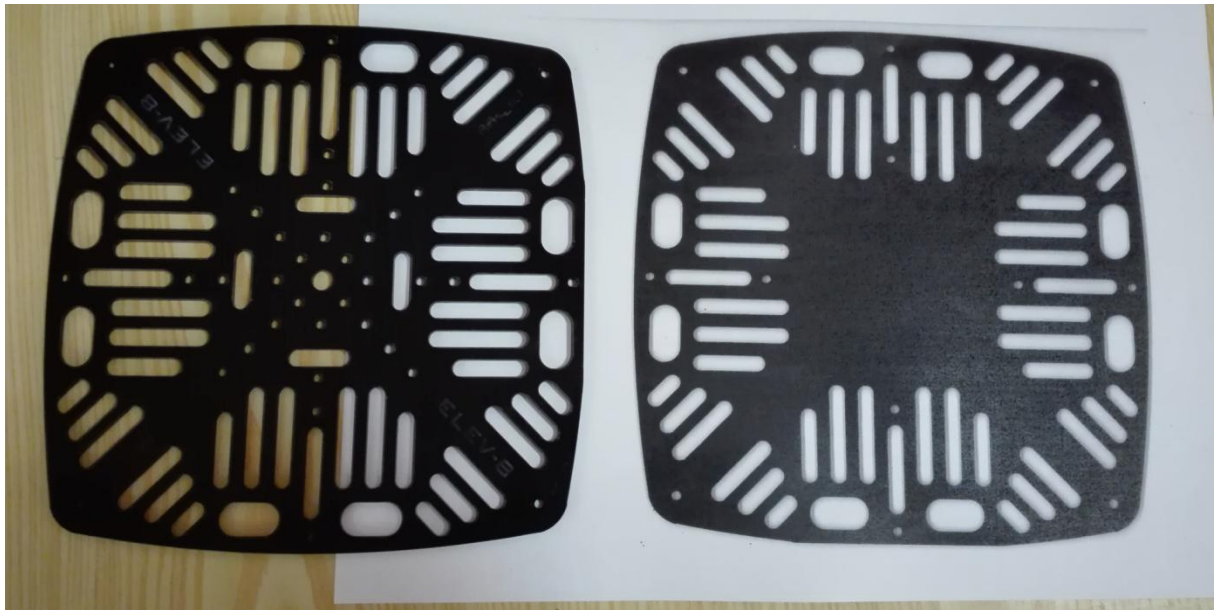
Et sur le deuxième(2) les batteries qui alimenteront le drone et le premier étage.

On a choisi cette méthode car sur un seul étage la raspberry ne pas rentrer avec les deux batteries.

De plus elles risqueraient d'endommager la raspberry en chauffant et dérégler le capteur thermique qui nous donnerait de fausses mesures.



Pour réaliser cette plaque nous avons demandé l'aide de la classe d'Itec du lycée qui nous a demandé de scanner la plaque à copier en taille réel puis de leur donner la photocopie.



Le résultat obtenu est celui ci-contre.

Du contre plaqué a été découpé pour avoir la forme de la plaque voulue et des vis ont été disposés au centre pour venir clipper la raspberry.



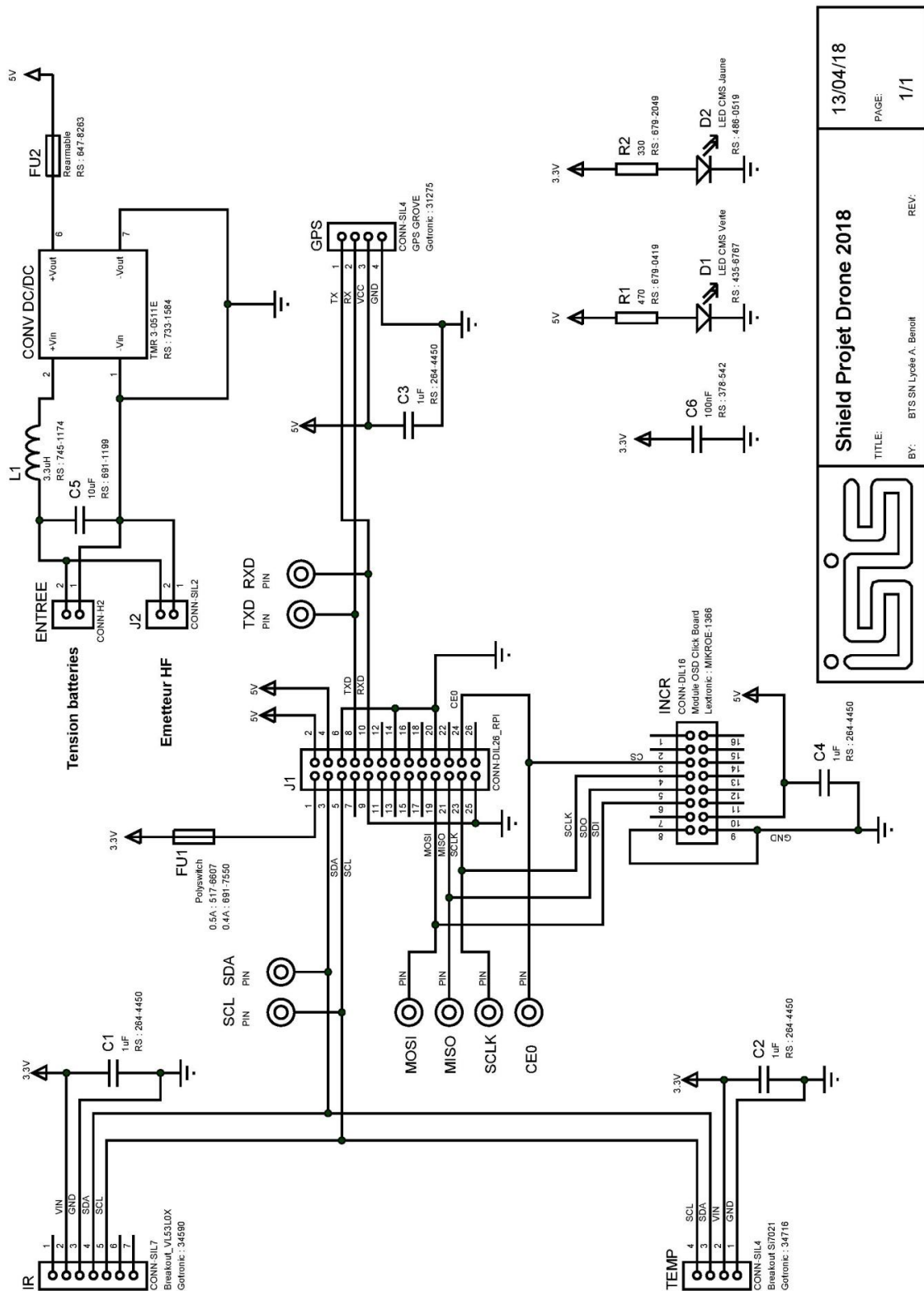
Annexes 1 :**Journal de bord :**

<i>Date</i>	<i>Objet</i>
11/01/2018	• Réunion début de projet
12/01/2018	<ul style="list-style-type: none"> • Comparaison du VL53L0X avec d'autre module infrarouge Réalisation fiche de comparaison avec un autre capteur infrarouge Prise de connaissance des librairies arduino et rpi <ul style="list-style-type: none"> • Planification des tâches
17/01/2018	• Test du VL530X sur la carte Arduino Uno
•18/01/2018	• Test des différentes librairies pour rpi
•19/01/2018	• Schéma Fritzing
•24/01/2018	• rédactions d'une fiche d'utilisation du VL53L0X et de cmake
•31/01/2018	• Temporisation de la mesure, analyse de trame, amélioration du programme
•08/02/2018	• Rédaction de document
•15/02/2018	• Etudes de la disposition et des dimensions du shield sur le drone.

●28/03/2018	● Scan de la plaque du drone, finalisation de la V1 du routage.
●29/03/2018	● Correction du routage sous Ares, recherche du positionnement des composants.
●04/04/2018	● Conception du shield de BUFEE .
●05/04/2018	● Bon de commande, début du test sur prototype du shield de BUFEE
●06/04/2018	● Conception du prototype shield (perçage)
●11/04/2018	● Suite et fin Conception du prototype du shield
●12/04/2018	● Travail sur la partie physique.
●13/04/2018	●Test et correction du shield test qui fonctionne.

Annexes 2 :

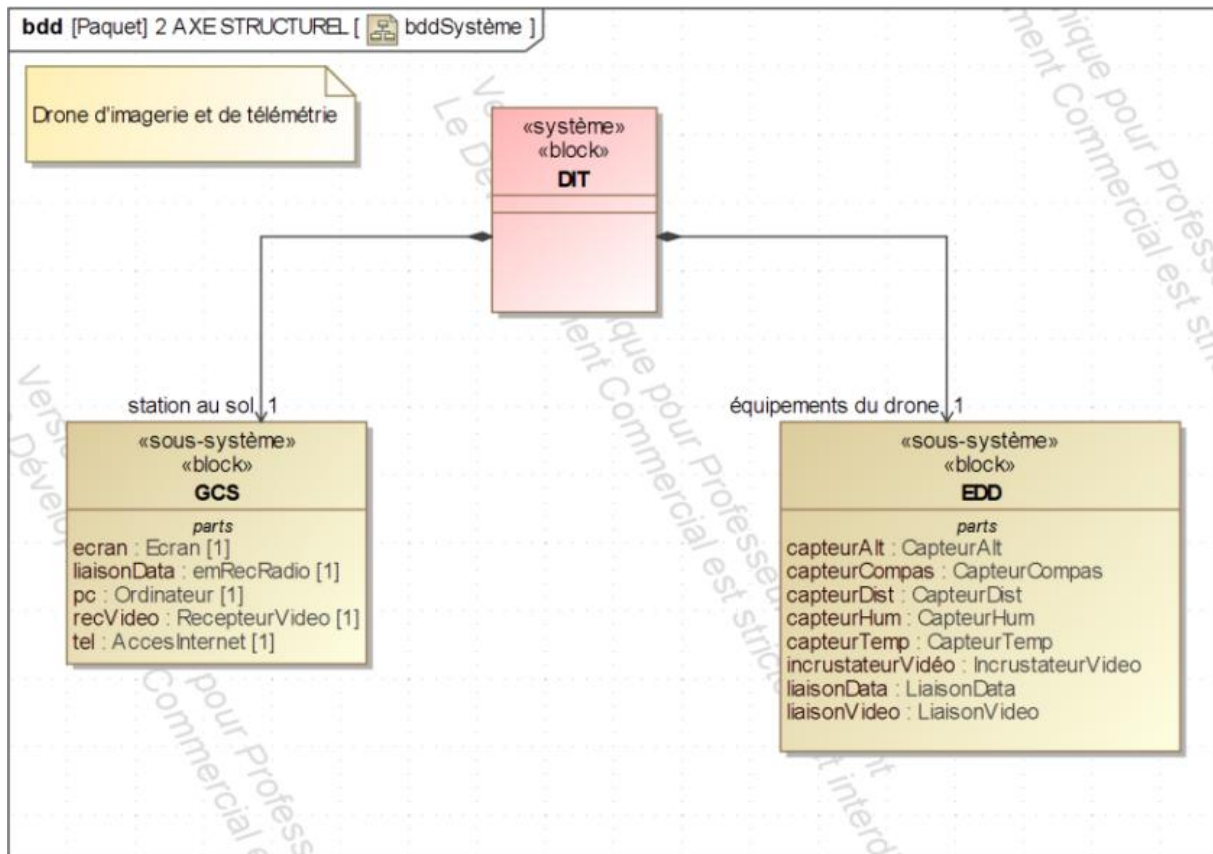
Schéma ISIS :



	Shield Projet Drone 2018 TITLE: BY: BTS SN Lycée A. Benoit REV:	13/04/18 PAGE: 1/1
---	---	--------------------------

Annexes 3 :**Gantt :**

▴ Tâches Bottigliero	184 h	Jeu 11/01/18	Mer 16/05/18		BOTTIGLIERO Nicolas
Recherche documentation + solution de l'année précédente	8 h	Mer 17/01/18	Jeu 18/01/18		
Test des différent librairie	10 h	Jeu 18/01/18	Jeu 25/01/18	43	
Test du capteurs sur Arduino et Raspberry	12 h	Ven 26/01/18	Jeu 01/02/18	44	
installation librairie + test du programme et amélioration	30 h	Ven 02/02/18	Mer 21/02/18	45	
Recherche de solution shield	20 h	Mer 14/03/18	Lun 19/03/18	46	
Routage	20 h	Mer 28/03/18	Jeu 05/04/18		
Travail sur la nomenclature	5 h	Jeu 05/04/18	Ven 06/04/18		
Fabrication d'une carte test	7 h	Ven 06/04/18	Mer 11/04/18		
test de la carte de test et modification en fonction	9 h	Jeu 12/04/18	Ven 13/04/18		
Travail de la partie physique	3 h	Ven 13/04/18	Ven 13/04/18		
reception de la carte et conception	8 h?	Jeu 11/01/18	Ven 12/01/18		
Finalisation	20 h	Jeu 10/05/18	Mer 16/05/18	48	

Annexe 4 :**Diagramme de bloc :**

Annexe 5 :

Conception de la carte :

Voici les étapes dans lesquelles il est préférable de souder la carte :

Pour le projet nous avons utilisé un four à refusions.

- D'abord commencer par les composants en CMS qui sont les plus petits et plats, ils ne viendront pas nous gêner dans la mise en place des composants traversant.
- Dans le cas de notre carte, les composants CMS du côté TOP seront soudés en premier car la bobine est beaucoup plus haute et pourrait devenir embêtante lors de la remise au four pour souder la face Bottom.
- Après la face Top soudait les composants CMS de la face Bottom.
- Souder le connecteur femelle qui ira sur le GPIO de la raspberry, car il est le seul sur la face Bottom.
- Puis sur la face Top, souder les connecteurs qui viendront accueillir les différents composants du shield.
- Puis vient le tour des Pin teste, le port grove, ainsi que l'entrer des batteries et du connecteur pour l'antenne et finir par le fusible 1.

Annexe 6 :**En cas de problème :****Si la communication I2C ne fonctionne pas :**

Vérifier de ne pas avoir mis à l'envers le capteur de distance ou le capteur de température et que le bus I2C soit activé dans le raspberry.

Préférence de la raspberry pi → Interface → cocher I2C et redémarrer la raspberry.

Si la LED verte ne s'allume pas :

Regarder si le shield est bien alimenté ou si la masse du convertisseur DC/DC est bien soudé.

Si la raspberry reboot au démarrage :

Elle est sûrement sous alimentée, vérifier le convertisseur DC/DC et la source de tension.

Nomenclature :

Mise à jour :		1106/2018		1/1	
Créé :	BOLLIGIERO Nicolas	02/04/2018			
	Nom	Date	Folio		
bouq glove	Emposee C14 couvract	Summ		golonuc	
EMILREE	Bonjour S couvract	3-2mm		Radiosbrales	
T3	Euprase S couvract	5-24mm		Golonuc	
T4	Couccerent SX13	3-24mm		Golonuc	
COMA DC/DC	Couvcrassent DCD/DC			Radiosbrales	
Gps2	Mecque Gps2 Glove 1130S0003			Golonuc	
INICR	Inocine «oag clicc poard »			Irextoluc	
LEMB	Cebien quimrdé et de leubérane			Golonuc	
IR	Cebien de districe			Golonuc	
bin femelle	Euprase femelle SO couvract	5-24mm		Radiosbrales	
bin mâle	S Couccerent Mâle travarsant	5-24mm		Radiosbrales	
T1	Indicance de choc	3-3H	SO %	Radiosbrales	
LUS	Enspice CMS résumple	4A – 5A		Radiosbrales	
LUN	Polyawitch trippie résumple	200µw – 0.52P		Radiosbrales	
D5	FED CMS Janue	H2MU-C110		Radiosbrales	5015
D4	FED CMS VArde	H2MG-C110		Radiosbrales	5015
CE	Coudusagient CMS	100 µF	50 %	Radiosbrales	802
CE	Coudusagient CMS	10 nF	10 %	Radiosbrales	1508
C4 g C4	Coudusagient CMS	1 nF	80 %	Radiosbrales	802
R5	Résistance CMS	330 Ohms	1 %	Radiosbrales	1508
R4	Résistance CMS	410 Ohms	1 %	Radiosbrales	603
Rebère	Désignation du matériel	Valeur (Référence)	Tol%	Equivaleur	

Projet Drone S018

							COEFFICIENT		0.000
							TOTAL HT GROBAT		101.615
							TOTAL HT FENITE		101.615

Client :

PROJET DRONE: DEMONCHY **MAXIME BTS SYSTEMES** **NUMERIQUES**



I. IR : DEVELOPPEMENT DE **L'EQUIPEMENT DU DRONE**

Présentation personnelle

Dans le projet je suis l'**IR 3** et je suis en collaboration avec quatre EC, dont un qui s'occupe de l'incrustation vidéo de la caméra et les trois autres de la réalisation de la classe C++ de chaque capteurs. Ma partie consiste à créer le logiciel du drone (EDD). Mon but est de développer le logiciel EDD en C++ à l'aide de QT Creator, en utilisant des capteurs de température, distance et d'humidité connectés sur une carte Raspberry PI 3. J'ai pour but de mettre en œuvre une gestion multitâches (réalisé des threads pour le fonctionnement des capteurs). Ainsi qu'une zone de mémoire partagée accessible par tous les threads ; une sauvegarde des mesures dans un fichier local, un système de communication entre processus. Et pour finir mettre en place la gestion de la caméra avec en complément un logiciel EDD de test.

Conception des multithreads

Le logiciel EDD doit travailler avec plusieurs processus divers qui fonctionneront en simultanés. Pour lancer plusieurs processus en simultanés la conception des multithread est nécessaire. Le thread est un bloc de code au sein d'une application dont l'exécution est parallèle au thread principal qui représente le plus souvent l'IHM de l'application. Le thread est utilisé car il ne nécessite pas l'allocation de segment mémoire. Sa création est donc plus rapide, mais il dépend du processus principal et s'il cesse d'exister, les threads qui en dépendent cessent aussi. Voici la fonction qui sert à créer le thread :

```
CCapteurTemHum_SI7021_A20::CCapteurTemHum_SI7021_A20(QObject *parent, int noMesBase) :
    QThread(parent)
{
    mShm = new CSharedMemory(this);
    connect(mShm, SIGNAL(sigErreur(QString)), this, SLOT(onErreur(QString)));
    mShm->memoireAttacher();

    mI2c = CI2c::getInstance(this, '1');
    connect(mI2c, SIGNAL(sigErreur(QString)), this, SLOT(onErreur(QString)));
    m_fin=false;
    m_noMesBase = noMesBase;
    qDebug() << "Objet CCapteurTemHum_SI7021_A20 créé!";
}
```

Conceptions de la zone de mémoire partagé

La zone de mémoire partagé (SharedMemory) sert à plusieurs sources de mettre leurs ressources en commun. Sous QT Creator nous avons la possibilité d'utiliser la classe « *QSharedMemory* ». Dans notre classe « *CSharedMemory* » nous allons créer un objet et l'utiliser de la manière suivante :

- Instanciation de l'objet QSharedMemory.
- Création (la première fois) du segment de mémoire partagé (create()).
- Attachement du segment au processus en cours (attach()).
- Utilisation protégée de la mémoire (pointeur).
- Détachement du segment du processus en cours (detach()).
- Destruction de l'objet

Voici le code pour créer la SharedMemory :

```
int CSharedMemory::memoireAttacherouCreer()
{
    int res;

    attach();
    if(!isAttached()){
        res = create(5*sizeof(T_Mes));
        if (!res) {
            QString mess="CSharedMemory::attacherOuCreer Erreur de création de la mémoire partagée.";
            emit sigErreur(mess);
            return ERREUR;
        } // if res
    } // if isAttached
    attach();
    m_adrBase = (float *)data();
    return 0;
}
```

Version Académique pour Professeur Seulement
Le Développement Commercial est strictement Interdit

package Data [EDD]

Diagram illustrating the architecture of a system, likely a robot or a sensor network, showing the flow of data and control between various components.

The main components and their interactions are:

- AppEdd** (Yellow box): The central application layer, containing logic for capturing data, managing threads, and handling errors. It interacts with **CCommuniq**, **CSharedMemory**, **CControlerCamera**, **CDevicePiMax7456**, **CSerialPort**, **QThread**, **QSocket**, **CcapteurDht_VL53L0X**, **CcapteurHum_SHT021_A20**, **CcapteurTemp_SHT021_A20**, **CcapteurGps**, and **T_Mes**.
- CCommuniq** (Orange box): Manages communication between the application and the shared memory.
- CSharedMemory** (Orange box): A shared memory space for data exchange between different modules.
- CControlerCamera** (Orange box): Manages the camera's operation, including setting the camera mode and capturing images.
- CDevicePiMax7456** (Orange box): A device interface for the PiMax7456, handling data capture and error reporting.
- CSerialPort** (Green box): A serial port interface for data exchange with external devices.
- QThread** (Orange box): A thread management component for executing tasks in parallel.
- QSocket** (Orange box): A socket interface for network communication.
- CcapteurDht_VL53L0X** (Blue box): A DHT sensor module for measuring distance and temperature.
- CcapteurHum_SHT021_A20** (Blue box): A humidity sensor module for measuring relative humidity.
- CcapteurTemp_SHT021_A20** (Blue box): A temperature sensor module for measuring temperature.
- CcapteurGps** (Blue box): A GPS module for location tracking.
- T_Mes** (Grey box): A measurement data structure containing various sensor readings.

The diagram shows a complex network of data flow and control signals, indicating a highly integrated system architecture.

Version Académique pour Professeur Seulement
Le Développement Commercial est strictement Interdit

Connexion entre la Raspberry et la GoPro

La connexion entre la **Raspberry** et la **GoPro** se fait en WiFi, il nous suffit juste de renseigner sur la **Raspberry** le « SSID » et le « PASSWORD » de la GoPro.

SSID GoPro: « *GoProStsSn* »

PASSWORD: « *go_pro_hero* »

Une fois connecté la Raspberry n'as plus accès a internet sauf si on la branche en câble. A partir de maintenant il nous suffit juste d'utiliser des trame « HTML » soit en passant en ligne de commande et donc on doit utiliser la commande suivante « *GET /camera/PW?t=<PASSWORD>&p=%01 HTTP/1.0\$0D\$0A\$0D\$0A* » (commande pour démarrer la **GoPro**) soit par un navigateur internet et donc on doit utiliser la commande suivante « <http://10.5.5.9/bacpac/PW?t=<PASSWORD>&p=%01> » (même commande), pour commander la caméra a distance.

Sur la **GoPro** il existe plusieurs paramètre important pour l'envoi de trame de contrôle, pour les paramètres générale de la GoPro on doit avoir « *bacpac* » dans la ligne de commande ensuite pour arrêter ou démarrer la **GoPro** on doit rajouter « *PW* » pour power et ensuite *01* pour démarrer et *00* pour arrêter et aussi pour changer de mode avec *02*, pour les paramètres vidéo on met « *SH* » avec pour démarrer une capture avec *01* et pour l'arrêter, pour les paramètres caméra on doit avoir a la place de « *bacpac* », « *camera* » ensuite pour lancé la vidéo en direct on doit rajouter « *PV* » avec *02* pour **ON** et *00* pour **OFF**.

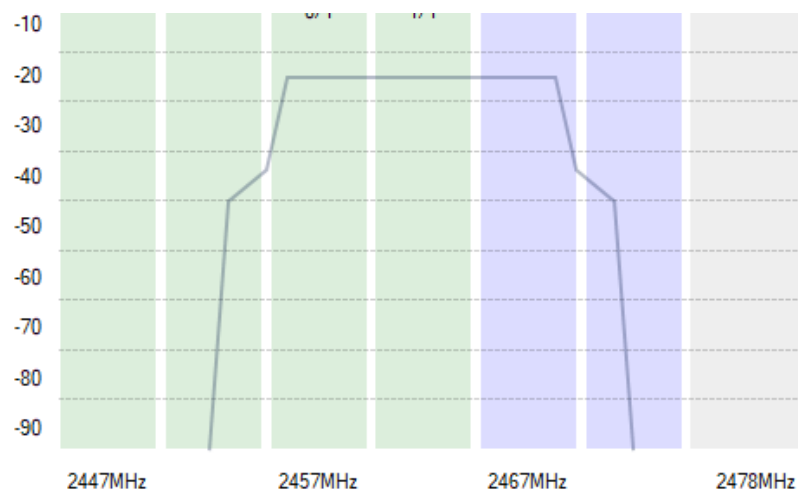
Présentation du WiFi de la Raspberry

Le WiFi est un ensemble de protocoles de communication sans fil régis par les normes du groupe IEEE 802.11. Un réseau WiFi permet de relier par ondes radio plusieurs appareils informatiques (ordinateur, routeur, Smartphone, modem Internet, etc.) au sein d'un réseau informatique afin de permettre la transmission de données entre eux.

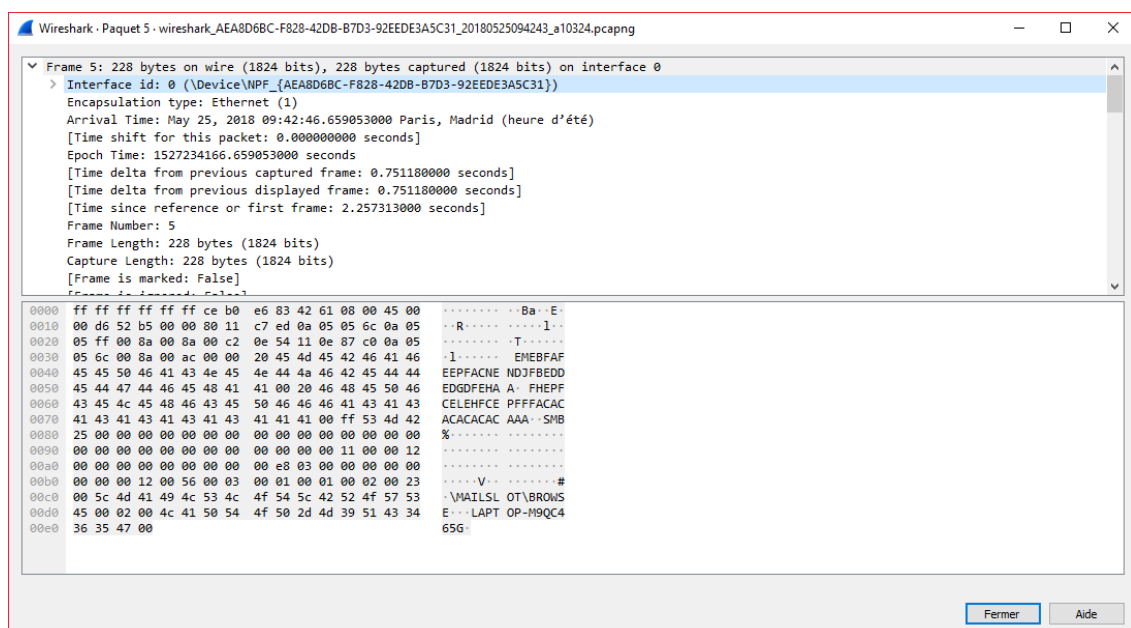
Le support WiFi 802.11.b/g/n est assuré par le chipset BCM43143 et l'antenne céramique est intégrée à la carte avec une fréquence de 2.4 GHz

Voici le spectre de fréquence de la connexion WiFi de la GoPro :

On peut voir la bande de fréquence entre 2450MHz et 2470 MHz.



Et voici l'exemple d'une trame d'envoi en hexadécimales entre un PC Windows et la GoPro en WiFi :



Problème rencontré

Lors de la mise en œuvre de mes classes j'ai rencontré un problème au niveau du nom de mes attributions par rapport aux deux autres informaticiens utilisant les données de mes capteurs.

Pour résoudre ce problème j'ai consacré deux heures sur le projet avec les deux autres informaticiens pour définir les noms des attributions de mes capteurs.

J'ai rencontré le problème d'utilisations des commandes en ligne de commande (« **GET** ») lors de l'envoi et de l'exécution. Et j'ai aussi rencontré le problème avec la commande **http** avec pour le password qui ne faut pas mettre entre "<>".

Semaine à venir :

Dans les semaines à venir je vais mettre en œuvre la connexion et le contrôle depuis la Raspberry vers la GoPro et mettre en œuvre les capteurs et l'incrustateur.