Baptiste GRILLERE
Olivier MAHIEUX

Projet IMA 4e année
Sonde Atmosphérique

2017

SOMMAIRE

Introduction

Dans le cadre du deuxième semestre de quatrième année d'école d'ingénieur, il nous a été proposé différents sujets de projets. Le but de ces 10 semaines de projet est de nous apprendre à réaliser un système informatique ou communicant afin de mettre en oeuvre nos connaissances acquises en IMA. Une démarche ingénieure sera donc à mettre en oeuvre afin de nous préparer à nos futurs stages et emplois.

La première étape a été de choisir notre sujet. Après lecture des propositions et questionnement sur le travail à réaliser, nous nous orientions sur des projets de systèmes physiques communicants et c'est la réalisation d'un ballon atmosphérique qui nous as motivé pour son aspect exploratif.

En effet, un ballon peut évoluer à des altitudes inhospitalières pour l'homme et seule la conception d'un système embarqué peut permettre l'étude de cet environnement.

I Présentation du projet

1) Contexte du projet

Afin d'acquérir des mesures de plusieurs grandeurs physiques à haute altitude (20 000m - 30 000m), le ballon atmosphérique fut largement utilisé dans la météorologie jusqu'à l'avènement de moyens plus modernes pour les prévisions météo, notamment avec la mise en orbite de satellites d'observation.

La demande croissante en objets connectés de ces dernières années a abouti au développement d'un réseau en capacité de répondre aux nouvelles exigences de ces systèmes. Le réseau LoRa (pour Low Range transmission network) permet la transmission de données à faible bande passante et de manière moins énergivore que d'autres moyens de communications.

Notre projet aura donc pour but de développer un ballon sonde atmosphérique utilisant le protocole LoRa.

2) Composition du système

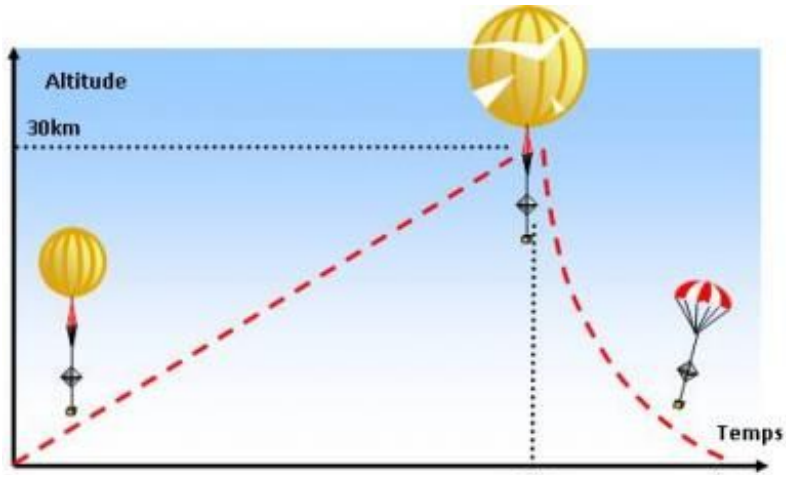


Schéma des différentes étapes d'un lâcher

<u>L'enveloppe:</u>

L'ascension est possible grâce à une enveloppe gonflée à l'hélium. Ce gaz, plus léger que l'air, réalise une poussée verticale. Avec le gain d'altitude, l'air devient de plus en plus rare et le gaz dans l'enveloppe se dilate donc avec la baisse de pression extérieure.

L’ascension se termine à l’éclatement de l’enveloppe aux alentours d’une trentaine de kilomètres (suivant le type d’enveloppe).

La nacelle:

Cette partie contient toute l’électronique de bord qui va permettre la collecte de données et la communication avec le sol. Elle va devoir aussi subir des contraintes de température mal supportées par les composants.

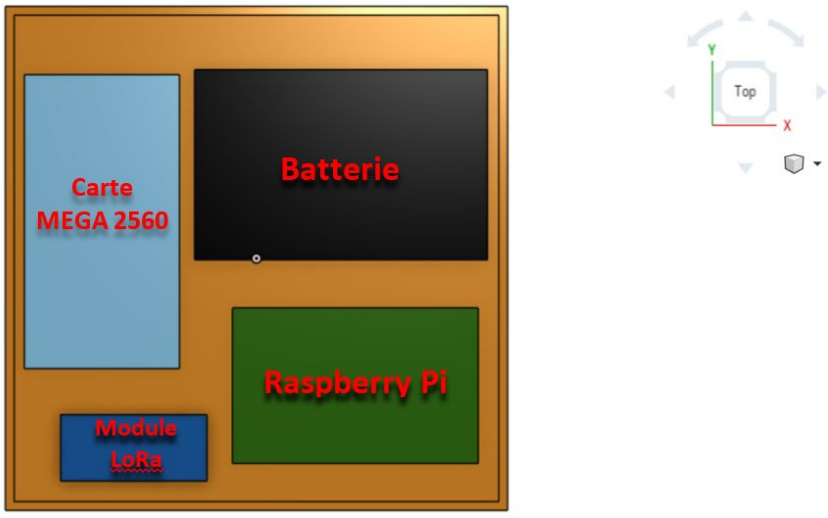
En effet, pour une altitude potentielle d’éclatement de 30 km, on peut observer des températures atteignants les - 50°C.

Pour solutionner ce problème, nous pouvons nous inspirer de projets déjà réalisés où l’on utilise largement du polystyrène comme isolant, ce qui limite les variations de températures et la perte d’énergie thermique, ainsi qu’une chaufferette chimique apportant de la chaleur à l’intérieur de la nacelle pendant l’ascension.

Par cette structure, il apparaît possible de conserver une température au dessus de 0°C tout au long du vol, ce qui reste acceptable pour l'électronique.

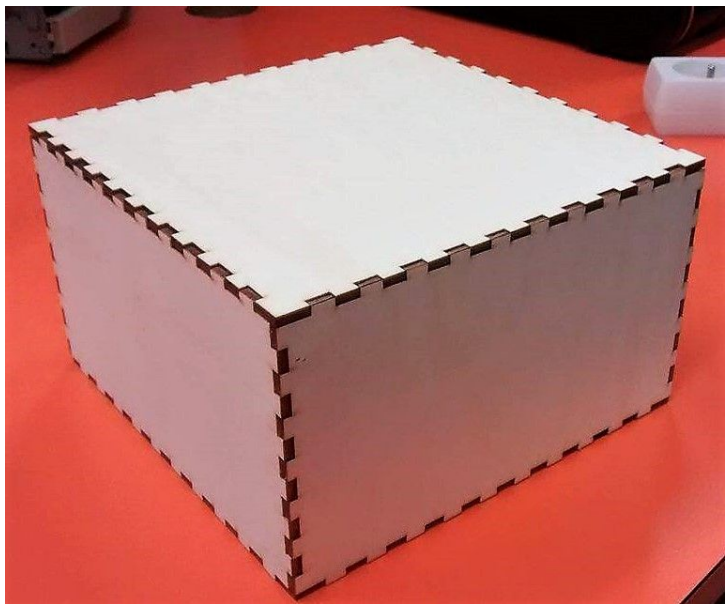
Dans l’optique d’un potentiel lâcher, nous avons conçu une nacelle en bois qui reste simplifiée (dans le sens où elle ne comporte pas de protection thermique).

Pour cela nous nous sommes servis du site de CAO libre OnShape pour modéliser les composants et ainsi dimensionner la nacelle à la bonne taille.



Disposition des éléments de la nacelle (vu du dessus)

Puis nous avons importé le schéma préalablement fait sur la découpeuse laser. L'avantage d'une telle conception reste le temps de réalisation: seulement 5 minutes sont nécessaires à la découpe.



Nacelle une fois assemblée

Parachute et réflecteur radar:

Lorsque l'enveloppe éclate et atteint donc son altitude maximale, la nacelle entame sa chute vers le sol. Sous l'action du flux d'air venant d'en dessous, le parachute se déploie naturellement et ralentit la vitesse de l'ensemble.

Le réflecteur radar doit permettre aux avions alentours d'éviter le ballon pour éviter tout incident.

Pour raison de non-lancement de notre ballon (justifié par la suite dans la partie législative), nous n'avons pas étudié ces deux éléments durant ce projet.

3) Législation

Le lancement de ce type de dispositif nous a demandé un rassemblement d'informations sur la législation. En effet, un ballon atmosphérique atteint une hauteur de plusieurs dizaines de kilomètres et entre dans la zone de l'aviation civile. De plus, la chute libre d'un objet depuis cette altitude suite à un problème (non-déclenchement du parachute) peut causer de gros dégâts au sol. Enfin le point de chute est aléatoire et peut par exemple causer un arrêt des transports si ce dernier tombe sur une ligne ferroviaire électrifiée ou une piste d'aéroport.

Pour obtenir ces informations nous avons contacté l'association Planète Sciences située à Denain et habituée au lancement de ballon atmosphérique dans le cadre de projets éducatifs. Ces derniers nous ont proposé de réaliser un lancement dans les conditions de vol de planètes science. Cependant cela impliquait l'utilisation de leur système qui ne comprenait pas de communication LoRa. Nous ne respections donc pas notre cahier des charges.

Afin de couvrir ces risques, des assurances conséquentes sont nécessaires pour couvrir les dommages. De telles assurances ne peuvent être délivrées à un particulier et possèdent des tarifs très élevés. Polytech ne pouvant nous couvrir, il nous est impossible de faire décoller notre ballon sans enfreindre la loi.

Dans le cas où l'on possède ces assurances, la démarche est la suivante :

En premier lieu, il faut obtenir les autorisations nécessaires auprès de l'aviation civile et notamment une demande NOTAM avisant les navigateur aériens de l'envoi du ballon. Cette demande doit être réalisée au moins 30 jours à l'avance.

En relation avec notre situation géographique la même démarche est à faire auprès des pays voisins, Belgique et Pays-Bas.

Les fréquences de communications doivent être différentes des fréquences utilisées par l'état afin de ne pas brouiller les communications, une utilisation dans les bandes de fréquences 400 MHz est donc adaptée.

Une description détaillée des conditions de lancements est décrite dans le document CELEX 32012R0923 appendice 2 page 54 à 58.

4) Cahier des charges

- Electronique embarquée :

- Relevé en temps réel des températures intérieures et extérieures. Capteur de température .
- Relevé de pression en temps réel.
- Relevée de l'altitude grâce au GPS.
- Transmission des données par technologies LoRa (température, pression, vitesse, altitude)
- Mise en place d'un système embarqué Raspberry pour coordonner le fonctionnement.
- Additionnel : Utilisation de la caméra de la Raspberry afin de prendre quelques photos lors du vol. Stockage des images sur carte SD.
- Alimentation du système par batterie.

- Structure de la nacelle :

- Utilisation de chaufferettes chimiques pour protéger la partie électronique sensible aux faibles températures.
- Structure en polystyrène de la nacelle afin d'alléger et d'isoler le système.

II) Partie technique

1) Disposition matérielle:

L'ensemble de notre dispositif électronique embarqué tourne autour d'une Raspberry Pi. C'est elle qui fait la liaison entre les données acquises et le module Lora qui va les envoyer au sol. Elle gère donc le traitement des informations GPS et des capteurs.

La configuration de la Raspberry a été la première tâche à réaliser. Nous avons mis en place la connection ethernet pour la programmer à distance par protocole ssh. De cette façon les ports séries ont été libérés et les appareils branchés de cette manière.

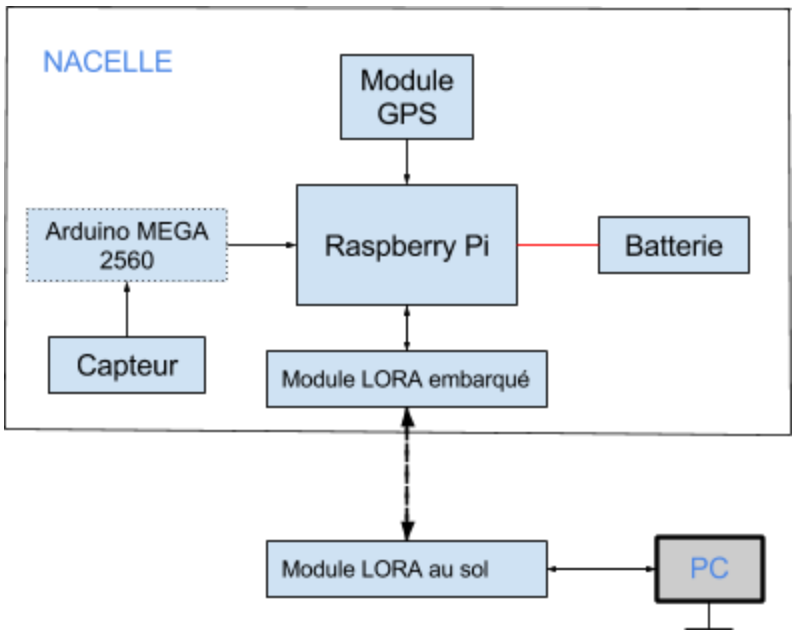
Le GPS et le module LoRa sont branchés sur les ports USB série.

La batterie fournie l'alimentation par le port Micro-USB.

Les capteurs initialement prévus pour être connectés directement sur le GPIO de la Raspberry par I2C ou SPI sont finalement délocalisés sur une carte Arduino branchée sur un port USB: cette carte Arduino n'était pas attendue dans le cahier des charges car elle consomme plus d'énergie que prévu. Elle est pour l'instant l'alternative à l'acquisition SPI complexe à réaliser.

Un module LoRa au sol réceptionnera les données du module LoRa embarqué. Cela est possible grâce à l'interface UART du module *Feather*.

Schéma matériel de notre système:



2) Choix des capteurs

Le choix des capteurs est très importants pour notre problème, en effet nous atteignons des altitudes très haute et donc un milieu très éloigné des conditions standards habituelles.
Nous devons donc choisir pour cela des capteurs robustes, pouvant travailler à des températures faibles et possédant une très grande étendue de mesure.
Nous utilisions une RaspberryPi et nous avons aussi choisi ces capteurs pour leur sorties I2C/SPI.

Ci-dessous les caractéristiques principales résumées des capteurs :

- Capteur de pression HSCDAND030PGSA3 Honeywell

Table 1. Absolute Maximum Ratings[1]

Characteristic	Min.	Max.	Unit
Supply voltage (V_{supply})	-0.3	6.0	Vdc
Voltage on any pin	-0.3	V_{supply} + 0.3	V
Digital interface clock frequency: I²C SPI	 100 50	 400 800	kHz
ESD susceptibility (human body model)	3	—	kV
Storage temperature	-40 [-40]	85 [185]	°C [°F]
Soldering time and temperature: lead solder temperature (SIP, DIP) peak reflow temperature (SMT)	 4 s max. at 250 °C [482 °F] 15 s max. at 250 °C [482 °F]		

Extrait de la documentation du capteur

Ce capteur peut donc communiquer ses mesures de façons numériques par I2C ou SOPI. Nous avons installé les librairies de fonction pour I2C cependant nous avons rencontré un problème lors de la recherche du registre de communication.
Une commande *i2cdetect -y 0* permet d'obtenir les registres I2C actifs du capteur.
Le choix du registre est renseigné dans la datasheet. Ainsi on peut fixer la liaison I2C sur le bon registre de communication.
Notre commande I2C indique ceci :

```
root@raspberrypi:/home/pi/code# i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- 06 -- 08 09 0a 0b -- 0d 0e 0f
10: 10 -- 12 13 14 15 -- 17 18 19 1a -- 1c 1d 1e 1f
20: -- 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f
30: 30 31 32 33 34 35 36 37 38 39 3a 3b -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
root@raspberrypi:/home/pi/code#
```

commande affichant les différents registres sur le bus I2C

Nous avons donc plusieurs registres probables et non fixes lorsque l'on répète la commande. Aucune information n'est pourvue dans la documentation pour préciser le registre à utiliser.
La connection SPI ne simplifiant pas les choses, une carte arduino à été provisoirement intégrée dans la nacelle afin d'acquérir les données de pression en analogique. C'est un arduino Mega donc surdimensionné par rapport à l'utilisation. Un Arduino Micro ou Nano aurait été mieux dimensionné mais n'étaient pas disponibles.
Ce dispositif consomme en revanche plus de puissance et réduit l'autonomie de la nacelle.

- Capteur de température TMP102AQDRLRQ Texas Instrument

1 Features

- AEC-Q100 Qualified with:
  - Temperature Grade 1: –40°C to 125°C Ambient Operating Temperature Range
  - HBM ESD Classification Level 2
  - CDM ESD Classification Level C6
- SOT563 Package (1.6 mm × 1.6 mm) is a 68% Smaller Footprint than SOT23
- Accuracy Without Calibration:
  - 2°C (Max) from –25°C to 85°C
  - 3°C (Max) from –40°C to 125°C
- Low Quiescent Current:
  - 10-µA Active (Max)
  - 1-µA Shutdown (Max)
- Supply Range: 1.4 V to 3.6 V
- Resolution: 12 Bits
- Digital Output: SMBus™, Two-Wire, and I^2C Interface Compatibility
- NIST Traceable

Extrait de la documentation capteur

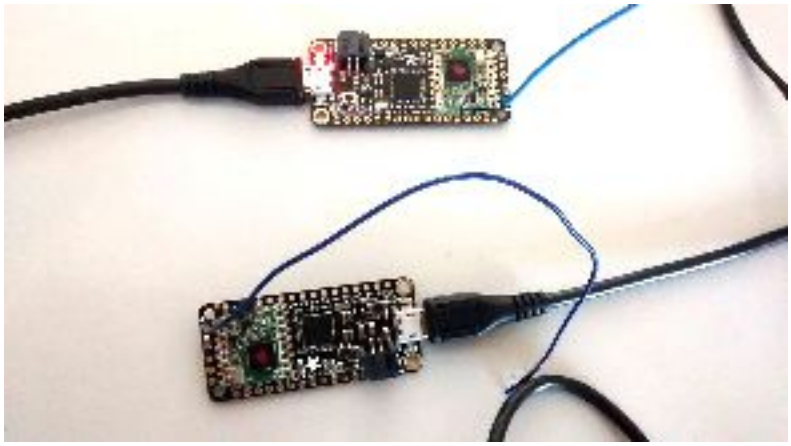
Ce capteur fonctionne en I2C ou en SPI mais pas en analogique. Nous n'avons pas pu l'utiliser car nous nous concentrions sur le capteur de pression exploitable.

3) Technologie LoRa

La technologie LoRaWAN (Long Range Wide Area Network) est actuellement en fort développement. Elle permet en effet de créer un Réseau d'objet connecté à très faible consommation d'énergie.
Le réseau LoRaWAN utilise le réseau de télécommunication en partenariat avec de grand opérateurs comme Orange,Bouygues Telecom ou Swisscom et alloue une certaine bande de fréquence dans les 868MHz (pour notre module). Ainsi il possède une couverture très grande.
De même, le réseau LoRa possède une très grande bande passante. Cela permet de connecter un nombre important d'appareil mais réduit la puissance et la vitesse de transmission.

Choix du module *Feather :*

Cette carte possède plusieurs avantages. Premièrement elle est accessible par liaison USB et est compatible avec le logiciel Arduino IDE. La bibliothèque déjà existante RadioHead RFM9X permet une programmation simplifiée à partir des squelettes de code donnés dans la documentation du composant.



Les deux modules Feather (embarqué et au sol)

Il est possible de choisir entre deux types de communications avec le protocole Lora: D'une part, l'utilisation du réseau LoRaWAN qui est mis en place par différents opérateurs et qui couvre maintenant une grande partie du territoire Français et d'autres part, la transmission radio *RFM9x* interne par les deux antennes respectives des modules.

Le second choix est préféré puisque le premier nécessite un droit d'accès au réseau pour la fréquence de 868MHz tout en sachant que cela rendra la récupération éventuelle plus difficile étant donnée la portée limitée des antennes seules. Cela ne change cependant pas l'utilisation du protocole LoRa requise par le cahier des charges.

Disposition des modules dans notre projet:

Le module au sol et celui embarqué ont leur propre programme. Le premier (LoRa1 sur le schéma ci-dessous) affiche sur le PC ce que le LoRa reçoit de la nacelle avec la fonction *serial.print.* Il transmet aussi les requêtes provenant du PC afin de changer de mode dans la nacelle

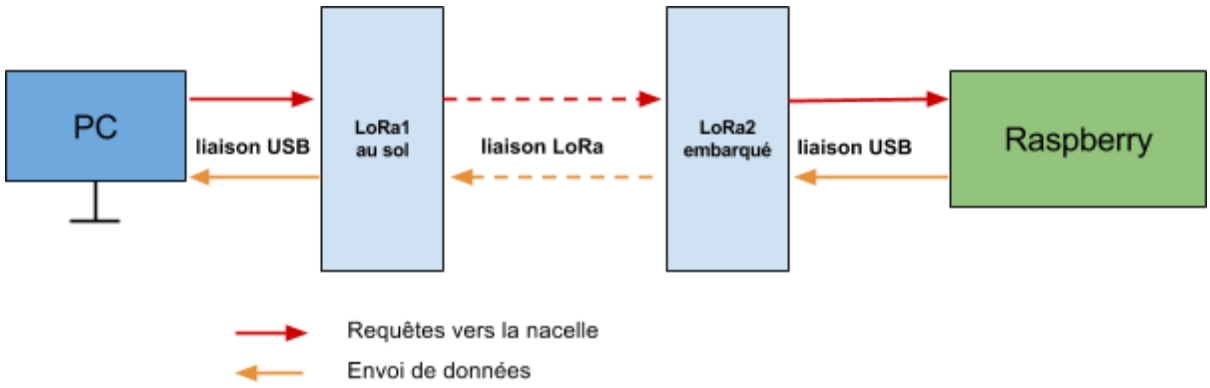
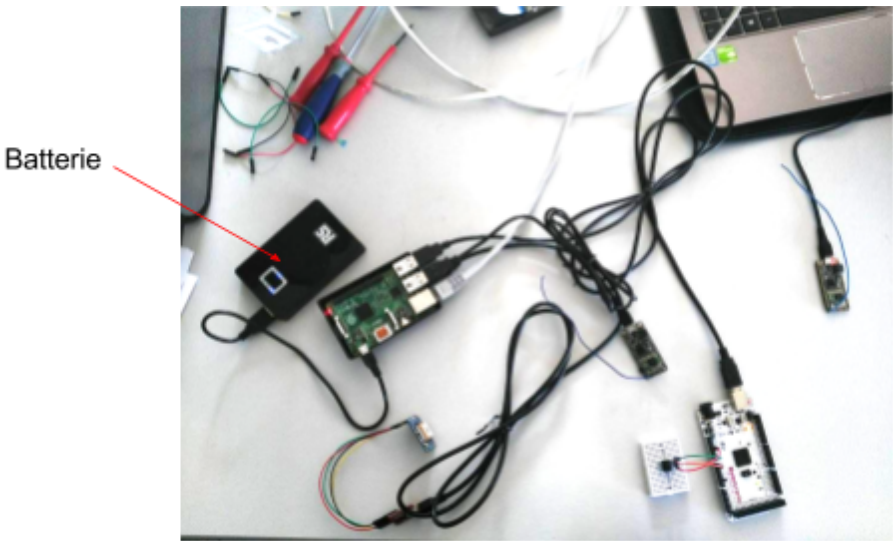


Schéma de la communication entre le sol et la nacelle

De la même manière, le second (LoRa2) se contente de transmettre sur sa liaison série ce qu'il reçoit par LoRa. Dans le sens inverse, il transmet au sol ce que la Raspberry Pi lui transmet sur la liaison.

4) Batterie



Localisation de la batterie parmi les élément de la nacelle

La batterie n'alimente directement que la Raspberry Pi puisque les autres éléments de la nacelle le sont par ses ports USB.

La batterie délivre 5A/h au maximum d'après la datasheet, la Raspberry et ses connections sont capables de consommer au maximum 2A/h. Notre système peut donc tenir 2h30 sur batterie en consommation maximum. La durée réelle est donc réduite car le

système ne consomme pas autant d'énergie. Un vol de ballon dure entre 2h et 3h ainsi un vol en autonomie est envisageable cependant une batterie plus puissante serait nécessaire si l'on compte le temps de récupération (pouvant potentiellement prendre quelques heures).

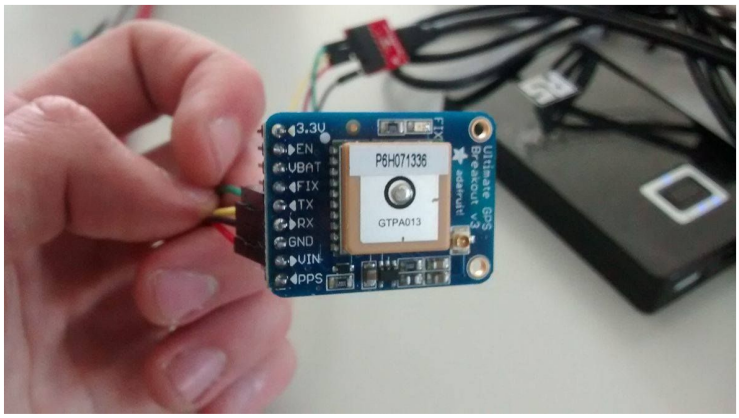
5) Module GPS

Le module GPS qui est embarqué dans la nacelle, doit pouvoir communiquer la position et l'altitude du ballon dans le but d'une récupération éventuelle. Il est connecté en USB à la RaspberryPi.

Le premier module GPS envisagé était un composant de la marque Maestro. Il avait l'avantage d'être intégré sur un PCB déjà conçu par un précédent projet.

Composant GPS Maestro

Malheureusement, après plusieurs tentatives de fonctionnement avec nos tuteurs, le module n'a jamais réussi à communiquer. Un nouveau module de chez Adafruit nous est alors fourni à la place.



Module GPS Ultimate Adafruit

Il est alimenté en 3,3V par la liaison USB de la Raspberry Pi avec laquelle il communique. Les 4 broches sont donc connectés (RX et TX pour la liaison série, la masse et la tension d'alimentation).

Il ne faut qu'une dizaine de secondes en extérieur pour que le module capte les satellites.

Norme NMEA:

Le *Global Positioning System* a été développé par la marine américaine et a été ensuite standardisé pour une utilisation plus grand public sous la norme dites *NMEA*.

En voici un exemple:

*$GPGGA,0640.36.289,4836.5375,N,00.740.9373,E,1,04,3.2,200.2,M,,,,,00,*0E*

```
$GPGGA       : Type de trame
064036.289   : Trame envoyée à 06h 40m 36,289s (heure UTC)
4836.5375,N  : Latitude 48,608958° Nord = 48°36'32.25" Nord
00740.9373,E : Longitude 7,682288° Est = 7°40'56.238" Est
1            : Type de positionnement (le 1 est un positionnement GPS)
04           : Nombre de satellites utilisés pour calculer les coordonnées
3.2          : Précision horizontale ou HDOP (Horizontal dilution of precision)
200.2,M      : Altitude 200,2, en mètres
,,,,,0000    : D'autres informations peuvent être inscrites dans ces champs
*0E          : Somme de contrôle de parité, un simple XOR sur les caractères entre $ et *[3]
```

Extrait de la page wikipédia de la norme

Une partie du programme embarqué dans la Raspberry Pi va donc traiter cette trame pour la découper et n'avoir uniquement que les informations utiles.

6) Raspberry Pi

Comme précisé précédemment, la RaspberryPi fait le lien entre les différentes données acquises dans la nacelle et la communication avec le sol via le protocole LoRa.
Ceci se fait à l'aide d'un programme mis en ouvre sur la carte et gérant les périphériques connectés en USB.

Deux versions de ce code sont existantes :

- La version manuel permet de commander les mode d'émissions de la nacelle sur requête de l'utilisateur au sol.
- La version automatique est utilisable pour un envol où la nacelle est complètement indépendante. L'utilisateur au sol est passif et reçoit simplement les données.

- Programme manuel

  Ci-dessous la listes des fonctions et leur fonctionnement :

  - Init_serial(char* device, int speed)

Cette fonction permet d'initialiser les ports USB de la Raspberry connectés à nos périphériques. La vitesse est fixés à 9600 Bauds. L'ouverture du file descriptor possède un argument important: O_NONBLOCK. En effet, à chaque appel de la fonction lecture_LORA(), le programme attendait une donnée sur la liaison USB. Cependant comme nous émettions de façon asynchrone sur la liaison, la totalité du programme se bloquait. Ce nouveau drapeau permet de continuer le programme dans le cas où rien n'arrive sur le port USB.

- Init_PORT()

Initialise les descripteurs de fichiers relatif à chaque port USB.

- check_mode()

Cette fonction appelée régulièrement dans le main vérifie les données reçues par le LoRa et les compare à des chaînes de caractères fixes (GPS, char, veille…). La valeur retournée permet d'entrer dans le mode correspondant ou bien de rester dans le mode actuel s'il n'y a pas de correspondance.

- lecture_LORA()

Écoute ce qui est présent sur le port USB LoRa et stocke la donnée dans la chaîne de caractère *reception*.

- lecture_GPS()

Écoute ce qui est présent sur le port USB GPS et acquiert les trames grâce à *fgets*. Ces trames sont ensuite analysée sémantiquement par la fonction sscanf afin d'obtenir les informations qui nous intéressent. Nous avons une vérification supplémentaire de l'acquisition correcte de ces trames avec le retour du sscanf. De plus nous avons une condition sur l'entête de la trame, en effet seules les trames de type *GPGGA* nous intéresse.

Les différentes valeurs de la trames sont envoyées,en fonction du mode en cours, au LoRa au sol.

- lecture_capteur()

Écoute ce qui est présent sur le port USB ArduinoMega et stocke la donnée dans la chaine de caractère *pression.* Cette donnée est ensuite envoyée au sol où elle est traitée par le LoRa terrestre.

- main()

Le programme tourne dans une boucle infinie.
Le mode initial est *en veille*. Le programme teste ensuite le mode dans lequel il se trouve. En fonction de celui-ci, il exécute les actions liées à ce mode ainsi que la fonction lecture_LORA(). La fonction lecture_LORA() inclut la fonction check_mode. A chaque appel de cette dernière, on vérifie donc le mode dans lequel nous nous trouvons. Le programme principal navigue donc entre ces modes.

- Code automatique

Dans ce programme, nous réutilisons les fonctions relatives à l'initialisation des ports, la lecture du capteur de pression et du GPS. De même, le code présente deux modes:

- le mode *Ascension* fait envoyer régulièrement les données de pression, l'heure précise et la position toutes les 10 secondes.
- le mode *Chute* envoie les mêmes données (sauf l'heure) toutes les secondes afin de connaître le plus précisément possible la position d'impact en cas de destruction du matériel.

Le programme se gérant sans requêtes extérieures, le changement de mode a lieu lorsque nous détectons un changement brusque d'altitude entre deux relevés (200 mètres en 10 secondes), indiquant la chute de la nacelle.

III) Essais de communication

Afin de pouvoir rester en contact le plus de temps possible avec la nacelle, la portée des radios du module LoRa doit être la plus grande possible.
La documentation nous indique une distance allant jusqu'à 2 km sans obstacles :

What ranges can I expect for RFM9X LoRa radios?

The RFM9x radios have a range of up to 2 km**line of sight** with tuned uni-directional antennas. Depending on obstructions, frequency, antenna and power output, you will get lower ranges - *especially* if you are not line of sight.

Nous vérifions donc cette donnée en éloignant au fur et à mesure les deux transmetteurs. La nacelle est en mode émission GPS (les données sont brutes, en norme NMEA).

```
Reception: 9862,N,00518.8717,E,0.52,0.85,1.26*06
$GPRMC,121547.000,A,4507.
RSSI: -81
Reception: $GPGGA,121548.000,4507.9862,N,00518.8717,E,1,9,0.84,349.3,M,48.6
RSSI: -77
Reception: ,24,15,12,10,32,17,25,19,,,,1.51,0.84,1.26*04
$GPRMC,121548.000
RSSI: -78
Reception: ,A,4507.9862,N,00518.8717,E,0.00,218.57,190417,,,A*60
$GPVTG,21
RSSI: -80
Reception: K,A*34
$GPGGA,121549.000,4507.9862,N,00518.8717,E,1,9,0.84,349.
RSSI: -82
Reception: 3,M,48.6,M,,*52
$GPGSA,A,3,13,24,15,12,10,32,17,25,19,,,,1.51,0
RSSI: -79
Reception: 7,33*70
$GPGSV,3,3,11,10,15,283,26,13,11,155,33,06,05,096,*4E
```

Capture du moniteur série du PC au sol

Nous observons une perte de contact à environ 500 mètres de distance, les deux modules n'étant plus en ligne de mire. Afin de caractériser ce décrochage, on s'intéresse au RSSI (Received Signal Strengh Indication) correspondant à la puissance en dBm du signal reçu. Le décrochage a lieu pour une valeur autour de -80 dBm.

Un deuxième essai est effectué en emmenant un des modules sur une colline environnante avec une vue dégagée afin d'être directement en ligne de mire.

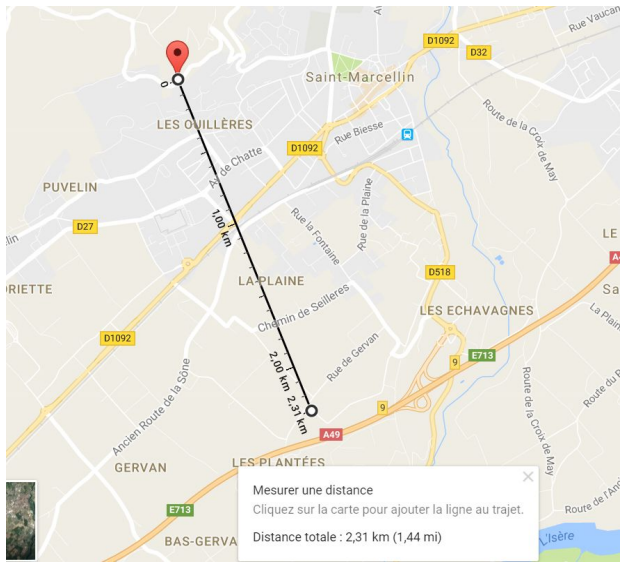
Placement des deux modules LoRa

Cette fois ci la nacelle est dans un mode d'envoi de caractères.

```
Reception: ggggggghe
RSSI: -90
Reception: -vvvvvvvv
RSSI: -85
Reception: pmmmmmmmmm
RSSI: -89
```

Capture du moniteur série

La réception fonctionne mais la qualité du signal dépend grandement de l'orientation de l'antenne, il faut se placer dans la bonne direction de polarisation. Le RSSI descend maintenant jusqu'à -90 dBm ce qui montre bien l'influence des obstacles. Nous validons ce que nous donnait la documentation puisque la portée limite est d'environ 2 km.



Mesure de la distance entre les deux modules

Conclusion

Ce projet comprend donc nos deux éléments, au sol et embarqué, en fonctionnement.

La communication LoRa est opérationnelle entre la nacelle et l'ordinateur au sol. L'équipement en vol comprend la Raspberry, la batterie, le GPS ainsi qu'un Arduino MEGA sur lequel est monté notre capteur de pression.

On a donc finalement 4 différences notables avec notre cahier des charges initial : le capteur de température manquant, l'utilisation d'un Arduino Mega, la présence d'un GPS opérationnel (une option du cahier des charges), l'absence de la gestion de la caméra.

La RaspberryPi centralise les périphériques et possède deux modes de fonctionnement : manuel et automatique permettant d'interagir ou non avec la nacelle.

Pour la puissance, notre batterie pourrait supporter un envol et un atterrissage cependant sans être sûr d'une émission continue au sol.

Ce projet nous a permis de nous investir en continu sur une longue durée. C'était un projet demandant l'application de nombreuses connaissances techniques apprises en IMA mais aussi de la gestion autonome de projet. Nous avons eu l'occasion de mener quelques projets auparavant et la démarche devient donc plus familière, un bon acquis pour la suite.

Bibliographie

Concernant la Raspsberry Pi et la liaison I2C:

-http://innovelectronique.fr/2013/03/02/utilisation-du-bus-i2c-sur-raspberrypi/
Utilisation du bus I2C pour Raspberry Pi, Electronique innovante

-https://learn.adafruit.com/adafruits-raspberry-pi-lesson-4-gpio-setup/configuring-i2c
Configuring I2C, Adafruit

Concernant la technologie LoRa:

-http://www.frugalprototype.com/technologie-lora-reseau-lorawan/
De la technologie LoRa au réseau LoRaWAN, Frugal Prototype

-https://www.raspberrypi.org/picademy/skycademy/
Skyacademy, raspberrypi.org

Concernant la législation:

-https://www.sia.aviation-civile.gouv.fr/reglementation
Réglementation, Direction générale de l'aviation civile

CELEX 32012R0923 appendice 2 page 54 à 58 (disponible sur le wiki).

Quelques prédicteurs de trajectoires:

-http://predict.habhub.org/

-http://astra-planner.soton.ac.uk/

Documentations des composants:

-https://cdn-learn.adafruit.com/downloads/pdf/adafruit-feather-m0-radio-with-lora-radio-module.pdf
Modules LoRa Feather

-https://cdn-learn.adafruit.com/downloads/pdf/adafruit-ultimate-gps.pdf
Module GPS

-http://docs-europe.electrocomponents.com/webdocs/11d9/0900766b811d9d41.pdf
Batterie

-http://www.mouser.fr/ProductDetail/Honeywell/HSCDAND030PGSA3/?qs=C%2f0tY%2f%2fqZe8RTnruveBFRQ==
Capteur de pression