



---

## Eirballoon 2 - Rapport de Stage

---

*Réalisé par:*

Asmae AAZOUZOU

*Encadrants :*

Guillaume FERRÉ

Bertrand LEGAL

## 1 Remerciements

Tout d'abord, je tiens à remercier mes encadrants, Monsieur Bertrand LEGAL et Monsieur Guillaume FERRE pour la confiance qu'ils m'ont accordée durant toute la durée de mon stage. Je les remercie particulièrement pour leurs précieux conseils, leurs bonnes orientations et pour le temps qu'ils m'ont consacrée.

J'adresse ensuite mes remerciements à Monsieur Guillaume DURIN, aérotechnicien chez l'AJ-SEP et chargé du suivi du projet pour son accompagnement, ses recommandations et ses précieux conseils.

Je tiens à remercier également les membres de l'équipe avec qui j'ai travaillé : M. Anthony GHIOTTO, M. Guy MORIZET, pour avoir fait de ce stage une expérience à la fois agréable et enrichissante.

Enfin, je remercie le groupe du traitement de signal et Image de l'IMS pour leur accueil chaleureux et leur disponibilité en cas de problème.

## Table des matières

<b>1 Remerciements</b>	<b>1</b>
<b>2 Introduction</b>	<b>4</b>
<b>3 Présentation de l'organisme d'accueil :</b>	<b>5</b>
<b>4 Processus de transmission d'images :</b>	<b>6</b>
4.1 Etapes de transmission d'images : . . . . .	6
4.2 Choix de fréquence et déroulement de l'émission : . . . . .	7
4.3 Matériel utilisé : . . . . .	8
<b>5 Modélisation de la chaîne de communications réelle</b>	<b>10</b>
5.1 Paramètres de la chaîne de communications : . . . . .	10
5.2 Émetteur : . . . . .	11
5.2.1 Association bits->symboles : . . . . .	11
5.2.2 Symboles différentiels : . . . . .	11
5.2.3 Filtre de mise en forme : . . . . .	12
5.3 Canal de propagation : . . . . .	13
5.4 Récepteur . . . . .	13
5.4.1 Synchronisation temporelle . . . . .	13
5.4.2 Synchronisation fréquentielle . . . . .	14
<b>6 Simulations</b>	<b>15</b>
<b>7 Transmissions réelles d'une image avec les HackRFs (MATLAB)</b>	<b>15</b>
<b>8 Implémentation en C++ du bloc d'émission sur la Raspberry Pi</b>	<b>17</b>
8.1 Prise et stockage des images . . . . .	17
8.2 Ecoute des signaux perturbateurs . . . . .	18
8.3 Contrôle de l'étage d'amplification avec la raspberry pi . . . . .	19
8.4 Emission en code Morse de l'identifiant du club radio . . . . .	20
8.5 Emission de l'image . . . . .	20
<b>9 Mesures de puissance</b>	<b>21</b>
<b>10 Tests en conditions de vol</b>	<b>22</b>
10.1 Tests à vide . . . . .	22
10.2 Complexités d'alimentation du module . . . . .	23
10.3 Tests en basses températures . . . . .	24
<b>11 Tests de transmission sur longues distances</b>	<b>25</b>
<b>12 Conclusion</b>	<b>25</b>

**13 Annexes**

**26**

## 2 Introduction

Étant actuellement étudiante en deuxième année d'école d'ingénieurs en filière électronique, j'ai effectué au Laboratoire de l'Intégration du Matériau au Système (IMS) un stage d'application entre le 31 Mai et le 17 septembre dont l'objectif est la réalisation de la deuxième version du projet Eirballon en partenariat avec le CNES, l'AJSEP et Planète Sciences.

En effet, ce projet consiste à la réalisation d'un système de communication entre un récepteur positionné au sol et un émetteur placé dans un ballon sonde. Il est réalisé par une équipe de trois étudiants ingénieurs en filière électronique et comporte deux grandes parties. La première partie consiste à la réalisation des mesures atmosphériques à savoir la température, la pression et le rayonnement UV et à la collecte des données GPS en utilisant le protocole de communication LORA. Les mesures atmosphériques seront réalisées tout au long de la durée du vol et ont pour objectif l'étude de la variation des différents paramètres mesurés en fonction de l'altitude. Les coordonnées GPS quant à elles permettront de connaître la position du ballon ainsi que son altitude afin de faciliter les recherches du ballon une fois celui-ci retombé.

Quant à moi, j'ai été chargée de la réalisation de la deuxième partie que je vais décrire tout au long de ce rapport. Elle consiste à la conception d'un système émetteur/récepteur dont le rôle est d' assurer le traitement et la transmission des images. L'émetteur sera embarqué dans un ballon sonde. Il est composé principalement d'une caméra usb dirigée vers le sol qui sert à capturer régulièrement des images. Ces images seront transmises tout au long du vol. De plus, cette émission se fait à une trentaine de km d'altitude . Par conséquent, elle sera confrontée à des conditions climatiques extrêmes à savoir de très basses températures et pressions. Cependant, nous n'avons pas obtenu l'autorisation pour émettre sur la fréquence 1,278GHz. De ce fait, nous avons placé un bouchon de  $50 \Omega$  pour stopper l'émission.

Dans ce présent rapport, je vais d'abord présenter le processus de transmission des images. Ensuite, je décrirai la chaîne de communication. Enfin, je présenterai les résultats des tests réalisés avant le vol.

### 3 Présentation de l'organisme d'accueil :

Le laboratoire de l'Intégration du Matériau au Système (IMS, CNRS UMR5218) a été créé en 2007, par la fusion de trois unités de recherche bordelaises, (IXL, PIOM, LAPS). Il suit une stratégie scientifique commune de développement principalement centrée dans le domaine des Sciences et de l'Ingénierie des Systèmes, à la convergence des Sciences et Technologies de l'Information et de la Communication, et des Sciences pour l'Ingénieur. Ce laboratoire est rattaché à trois institutions qui sont : le CNRS, l'Université de Bordeaux et Bordeaux Aquitaine INP.



FIGURE 1 – Le laboratoire de l'Intégration du Matériau au Système (IMS)

L'IMS compte actuellement environ 350 personnes : 135 chercheurs et enseignants-chercheurs, 150 doctorants et post doctorants, et 65 ingénieurs et techniciens. Le budget annuel d'IMS incluant les salaires est de 20 millions d'euros. Au cours des cinq dernières années, les chercheurs d'IMS ont publié 800 articles de revues, 1200 communications et déposé 50 brevets.

L'IMS compte aussi 10 groupes de recherche qui sont décrit dans la figure ci-dessous :

10 Groupes	28 Equipes			
BIOELECTRONIQUE N. Lewis	Bio-EM Y. Percherancier	ELIBIO Y. Bornat	AS2N S. Saighi	
COGNITIQUE J-M André	CIH J-M André	ERGO J. Petit	RUDII A. Lehmans	PMH_DySCo L. Arsac
SIGNAL Y. Berthoumieu	MOTIVE J-P. Da Costa	SPECTRAL A. Giremus		
PRODUCTIQUE Y. Ducq	MEI M. Traoré	ICO C. Merlo	LOCO2 R. Dupas	
AUTOMATIQUE X. Moreau	ARIA D. Henry	CRONE P. Melchior	FFTG F. Cazaurang	
CONCEPTION D. Dallet	CAS F. Rivet	CSH N. Deltimple	CSN C. Jego	
FIABILITE G. Duchamp	PACE H. Frémont	PIUSSANCE J-M. Vinassa		
NANOELECTRONIQUE P. Mounaix	LASER J-P. Guillet	MODEL C. Maneux	III-V N. Malbert	
ORGANIQUE I. Dufour	ELORGA M. Abbas	PRIMS C. Ayéla		
ONDES Y. Ousten	MDA C. Dejous	EDMINA L. Béchou	MIM F. Demontoux	

FIGURE 2 – Organisation scientifique de l'IMS

En ce qui concerne mon stage, il s'est déroulé au sein du Groupe Signal et Image (GSI). Ce

groupe s'occupe de la conception d'algorithmes pour le traitement avancé du signal au sens large. Les travaux menés, où modélisation, inférence, optimisation et échantillonnage sont omniprésents, concernent des sujets impliquant des aspects multi-échelles, multi-dimensionnels, statiques ou dynamiques. Les thématiques, sur lesquelles le GSI s'est engagé au cours du dernier quadriennal, visent la résolution de problèmes inverses notamment dans le cas de données incomplètes, la caractérisation, le filtrage, la prise de décision optimale et la transmission de l'information.

## 4 Processus de transmission d'images :

### 4.1 Etapes de transmission d'images :

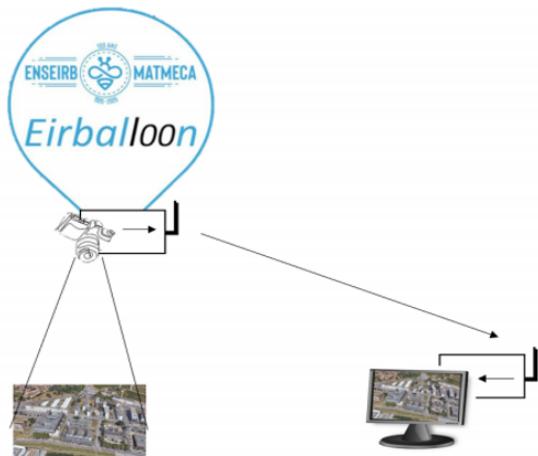


FIGURE 3 – Transmission d'une image depuis le ballon sonde

Comme il a été évoqué précédemment, notre système se compose d'un émetteur mobile situé dans le ballon sonde et d'un récepteur fixe placé dans le club radio de l'Enseirb-Matmeca. L'émetteur est composé principalement d'une caméra USB dont le rôle est de capturer sur un intervalle régulier des images du sol. Tout au long du vol ces images seront stockées dans un raspberry pi puis seront traitées avant d'être transmises par le système émetteur. Le système récepteur quant à lui reçoit les données les traite puis les stocke. Ci-dessous une représentation générale du processus de transmission des images :

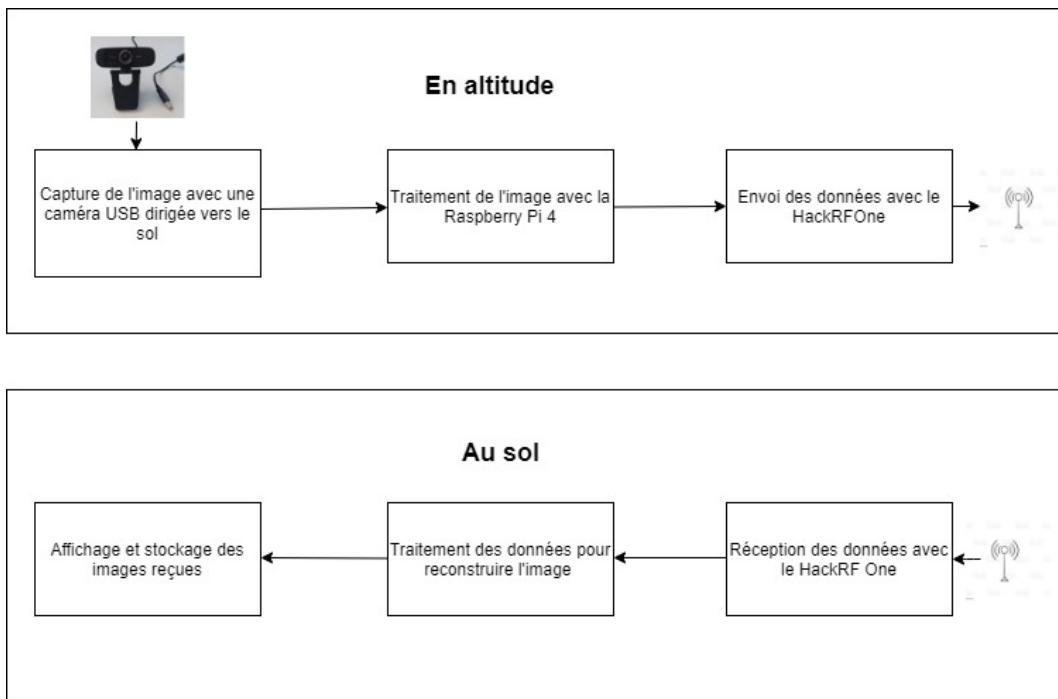


FIGURE 4 – Les étapes de la transmission d'une image

## 4.2 Choix de fréquence et déroulement de l'émission :

On a choisi dans un premier temps une fréquence d'émission qui vaut 1.255 GHz, Cependant nous étions obligés de la changer car plusieurs transmissions utilisent cette fréquence ce qui va perturber notre émission. Par conséquent, la nouvelle valeur sélectionnée est 1,278 GHz sur laquelle il n'y avait pas beaucoup de perturbations. Cependant, lors du lancé du ballon nous avons placé un bouchon de  $50 \Omega$  pour stopper l'émission car nous n'avons pas obtenu l'autorisation pour émettre sur cette fréquence

De plus, pour éviter ces perturbations nous disposons de deux antennes l'une sert à l'écoute et l'autre à l'émission. Avant chaque émission l'antenne d'écoute vérifie pendant une seconde s'il y a des signaux perturbateurs. Si c'est le cas, le processus d'écoute est relancé. Sinon, on envoie en morse l'identifiant du club radio puis l'image, par la suite on attend trois minutes avant de relancer le processus d'écoute. Ci dessous un schémas qui résume ces étapes :

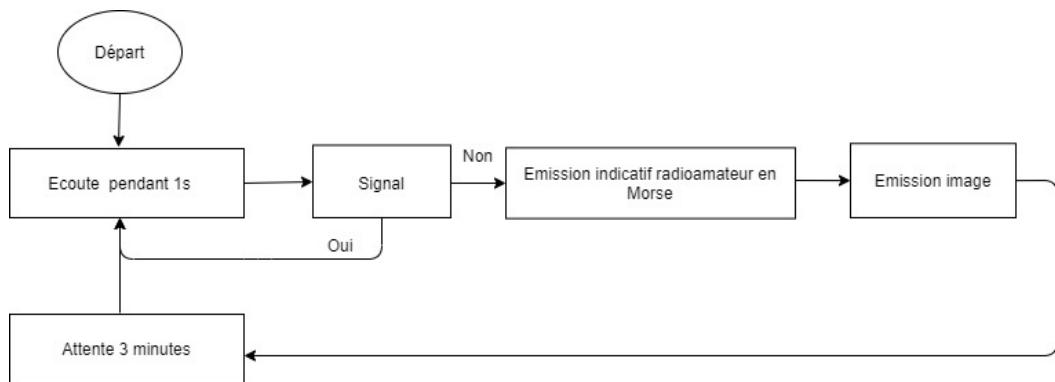


FIGURE 5 – Déroulement de l'émission

### 4.3 Matériel utilisé :

Pour concevoir le système d'émission qui sera chargé de transmettre les images, plusieurs outils ont été utilisés :

- **Une caméra USB** de type Spedal Full HD Webcam 1080p, fixé au-dessous de la nacelle. Son rôle est de capturer les images du sol.

- **Une carte Raspberry Pi 4 B** permettant le stockage et le traitement des images. Elle est dotée d'un processeur ARM Cortex-A72 64 bits ; quatre coeurs de 1,5 GHz et 8 GB de mémoire RAM et offre des performances de bureau comparables à celles des systèmes PC d'entrée de gamme x86. Cette dernière nécessite une carte SD munie d'un OS et une alimentation 5 Vcc/-maxi 3 A via prise USB Type C.

- **Module SDR (Software Defined Radio) HackRF One** qui est un périphérique radio à définition logicielle capable de transmettre ou de recevoir des signaux radio de 1 MHz à 6 GHz. C'est une plate-forme matérielle open source qui peut être utilisée comme périphérique USB ou programmée pour un fonctionnement autonome.



FIGURE 6 – Module SDR HackRF One utilisé

- **Un switch HMC435AMS8G** : C'est un composant électronique qui a une entrée (RFC) et deux sorties (RF1, RF2). Il est commandé par deux niveaux logiques opposés en A et B (Figure 6), si A = 0 et B = 1 la sortie active est RF1. si A = 1 et B = 0 la sortie active est RF2. Ce composant permet de basculer vers l'antenne d'écoute ou vers l'antenne d'émission en l'absence de perturbations.

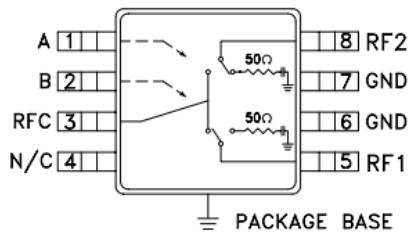


FIGURE 7 – Switch HMC435AMS8G

- **Un driver + amplificateur de puissance** : Il permettent l'amplification du signal à transmettre. Il fournissent 21 dBm de puissance
- **Deux antennes de 1.255 GHz Tx/Rx** : Une pour l'écoute (Figure 8 : antenne verte) et l'autre pour l'émission. (Figure 8 : antenne rouge)

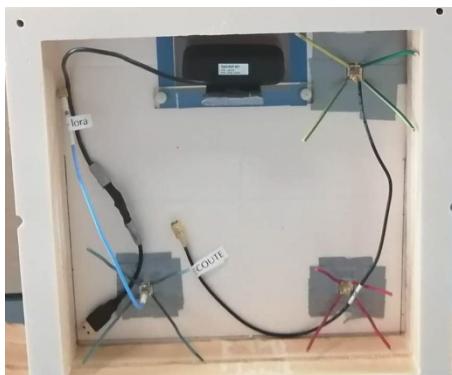


FIGURE 8 – Les antennes d'écoute et d'émission

- **Un régulateur 9-5 V + inverseur** : Le régulateur est commandé par une tension de 0-3V fournie par l'une des broches GPIO de la raspberry. Quand la raspberry délivre 3V alors la sortie du régulateur vaut 5V ; cette tension sera utilisée pour alimenter le driver, l'amplificateur de puissance (PA), et sera appliquée au pin B du switch. L'inverseur quant à lui va délivrer une tension de 0V qui sera appliquée au pin A ce qui permet de basculer à l'état de l'émission. Dans le cas contraire, le régulateur délivre 0V et on sera donc dans l'état de l'écoute. (Figure 9)

Les différents composants et la liaison entre eux sont bien détaillés dans le schéma suivant :

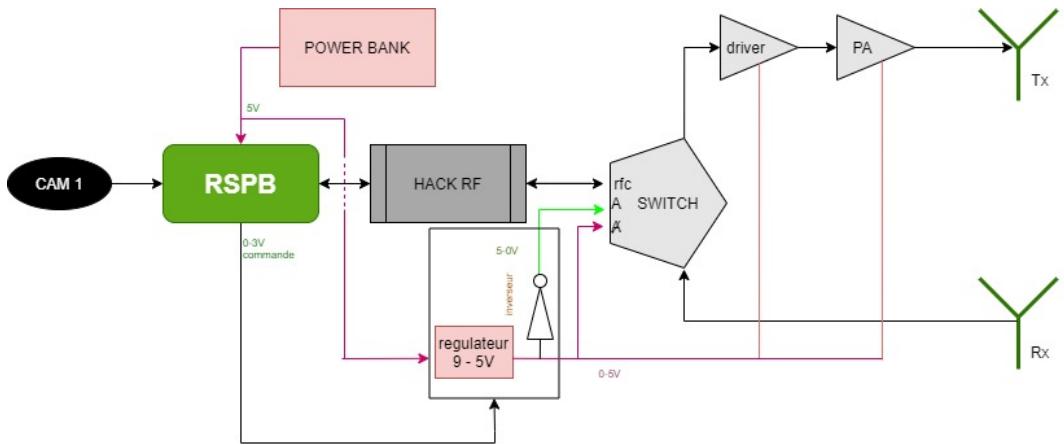


FIGURE 9 – Schéma complet du bloc de l'émission

Le système de réception est composé d'une antenne directionnelle montée sur rotor de type Yagi-Uda et un module SDR HackRF One. Le signal reçu sera décodé ensuite à l'aide d'un programme informatique.

## 5 Modélisation de la chaîne de communications réelle

### 5.1 Paramètres de la chaîne de communications :

Notre chaîne de communications est sur mono-porteuse. Ainsi, son architecture est semblable à :

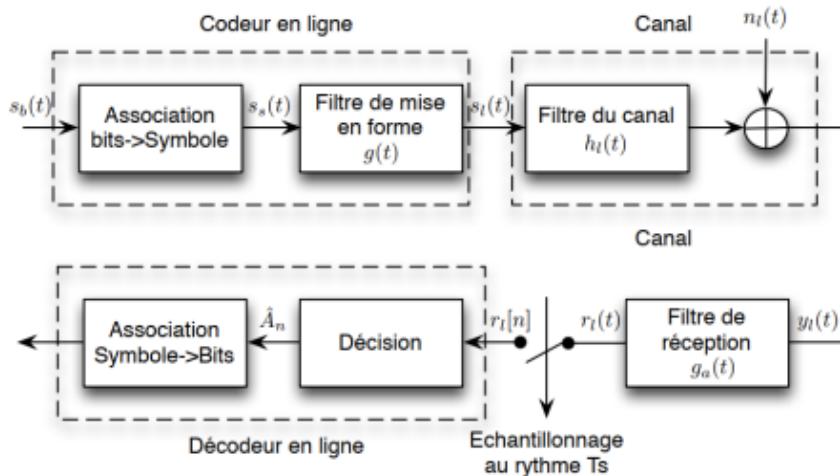


FIGURE 10 – Chaîne de communications numériques mono-porteuse

les paramètres retenus pour cette chaîne sont :

- type de la chaîne de communications : mono-porteuse
- modulation numérique : QPSK
- fréquence de la porteuse :  $f_p = 1,278 \text{ GHz}$
- puissance d'émission :  $P_e = 1 \text{ W}$

- débit binaire :  $D_b = 1 \text{ M bits/s}$   $D_s = 500 \text{ k Symboles/s} \Rightarrow T_s = 2 \mu\text{s}$
- facteur de suréchantillonnage :  $F_{se} = 6 \Rightarrow f_e = 3 \text{ MHz}$

## 5.2 Émetteur :

### 5.2.1 Association bits->symboles :

A l'entrée de l'émetteur, les canaux RGB de l'image sont récupérés. Ceux-ci forment un flux binaire présenté par l'équation suivante :

$$s_b(t) = \sum_{k=1}^{N_b} b_k \delta(t - kT_b) \quad (1)$$

Où  $T_b = 1 \times 10^{-6}s$  est la période binaire, et  $N_b$  est le nombre de bits. Comme l'image est codée en RGB et lue sous forme de uint8,  $N_b$  est égal au nombre de pixels de l'image multiplié par 24.

Avant de passer à l'étape de modulation numérique, on décompose l'ensemble de bits en plusieurs sous paquets au début desquels on insère l'octet de vérification "10101010" afin de vérifier le bon déroulement de la synchronisation en réception. En effet, lors de cette étape des erreurs peuvent survenir. Ils résultent d'éventuels décalages des paquets ; au cas où le décalage ne dépasse pas la moitié de l'octet, des corrections peuvent être mise en oeuvre pour décoder l'image convenablement.

Par la suite on procède à la modulation QPSK qui consiste à associer chaque couple de bits à un symbole comme suit :

Combinaisons de bits	Symboles QPSK
00	$e^{j\frac{\pi}{4}}$
10	$e^{j\frac{3\pi}{4}}$
11	$e^{j\frac{5\pi}{4}}$
01	$e^{j\frac{7\pi}{4}}$

FIGURE 11 – Association bits->symboles

### 5.2.2 Symboles différentiels :

Après l'association bits->symboles, on recourt à un codage différentiel qui permet de coder le symbole de rang n en fonction des symboles précédents selon l'équation récursive suivante :

$$S_d[n + 1] = S[n]S_d[n] \quad (2)$$

Avec  $n \geq 1$  et  $S_d[1] = e^{j\frac{\pi}{4}}$

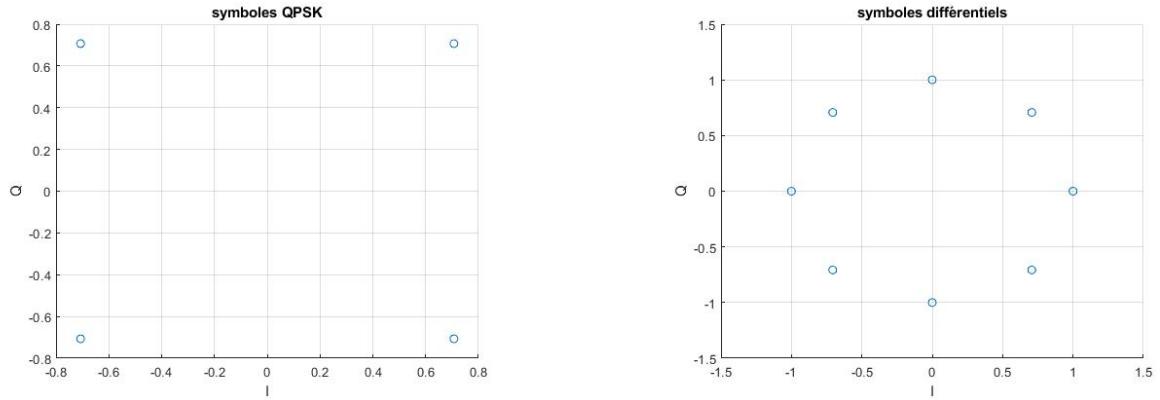


FIGURE 12 – Constellation I/Q des symboles QPSK sans et avec codage différentiel

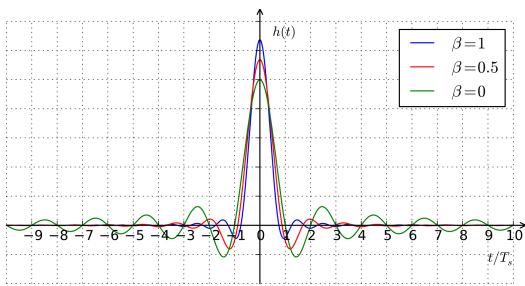
Ce codage est utilisé pour éviter (jusqu'à une certaine limite) les problèmes associés à un manque de synchronisation de phase entre l'émetteur et le récepteur. En effet, en comparant le premier symbole reçu avec le premier symbole théorique  $e^{\frac{j\pi}{4}}$ , on peut estimer le déphasage et le corriger.

Le signal à la sortie de modulateur numérique est représenté par :

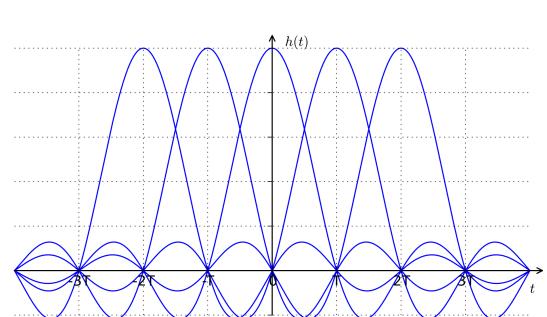
$$s_s(t) = \sum_{k=1}^{(N_b+8*nombre-de-paquets)/2+2} S_d \delta(t - kT_s) \quad (3)$$

### 5.2.3 Filtre de mise en forme :

La mise en forme du signal est réalisée avec un filtre en racine de cosinus surélevé. La convolution des deux filtres d'émission et de réception forment un filtre en cosinus surélevé qui est une implémentation d'un filtre de Nyquist ; par conséquent il a la propriété d'éliminer les interférences inter symboles (ISI) car sa réponse impulsionnelle est nulle tout les  $nT_s$  (où n est un entier), sauf n=0. (b)



(a) Réponse impulsionnelle du filtre en racine de cosinus surélevé pour trois valeurs de roll-off



(b) Réponse impulsionnelle du filtre à cosinus surélevé avec divers facteurs d'atténuation

FIGURE 13

On choisit un facteur de roll-off  $\beta = 0.8$ . Le signal de sortie du bloc du filtrage a l'expression

suivante :

$$s_l(t) = (g * s_s(t)) = \sum_{k=1}^{(N_b+8*\text{nombre-de-paquets})Fse/2+2} S_d g(t - kT_s) \quad (4)$$

avec

$$g(t) = \begin{cases} \frac{4\beta}{\pi\sqrt{T_s}} \frac{\cos \pi \frac{(1+\beta)t}{T_s} + \frac{\sin \pi \frac{(1-\beta)t}{T_s}}{\frac{4\beta t}{T_s}}}{1 - (\frac{4\beta t}{T_s})^2} & \text{si } -4T_s < t < 4T_s \\ 0 & \text{sinon.} \end{cases} \quad (5)$$

Pour des fins de synchronisation temporelle, on insère un préambule au début de chaque sous paquet formé avant de procéder à la modulation. Celle-ci s'effectue grâce au module SDR HackRF One qui fait monter Le signal  $s_l(t)$  à la fréquence porteuse. Cette étape génère un décalage fréquentiel  $\Delta f T_x$ , observé également au niveau de la réception, qui est dû à la différence entre la fréquence de l'oscillateur local de cette radio logicielle et la fréquence théorique de la porteuse  $f_p$ .

### 5.3 Canal de propagation :

La canal de propagation réel étant l'atmosphère, plusieurs contraintes doivent être tenues en compte lors de la transmission du signal :

- la présence d'un bruit : bruit thermique, des signaux perturbateurs...
- décalage temporel :  $\tau = \frac{R}{c} = 0,1s$  si on estime  $R_{max} = 30km$
- la perte de puissance du signal...

### 5.4 Récepteur

#### 5.4.1 Synchronisation temporelle

Le récepteur doit tenir compte de toutes les contraintes précédemment évoquées, notamment, le temps de propagation non nul du signal. Celui-ci est estimé grâce à la synchronisation temporelle basée sur la détection des préambules. Notons que la combinaison du préambule est unique et ne peut être retrouvée dans le flux de symboles issu de l'image. En effet, il est impossible de repérer à la sortie du filtre adapté des symboles nuls sur plusieurs périodes  $T_s$  successives. La méthode retenue pour la détection des préambules est l'intercorrélation entre le signal reçu et le préambule théorique. La formule de cette dernière est donnée en équation 6. En effet, la présence des pics excédents un seuil fixé à 0.8 reflètent le début de chaque sous paquet du signal.

$$\rho(\mu) = \frac{\sum_{n=1}^{N_p-1} y[(n + \mu)T_e]s_{preamble}[nT_e]}{\sqrt{E_y} \sqrt{E_{s_{preamble}}}} \quad (6)$$

où  $s_{preamble}$  est le signal associé au préambule de longueur  $N_p = [\frac{T_p}{T_e}]$  et  $T_p$  la durée du paquet. On effectue une transmission avec un nombre de paquets égal à 48, la figure ci-contre représente les 48 pics auxquels on s'attendait.

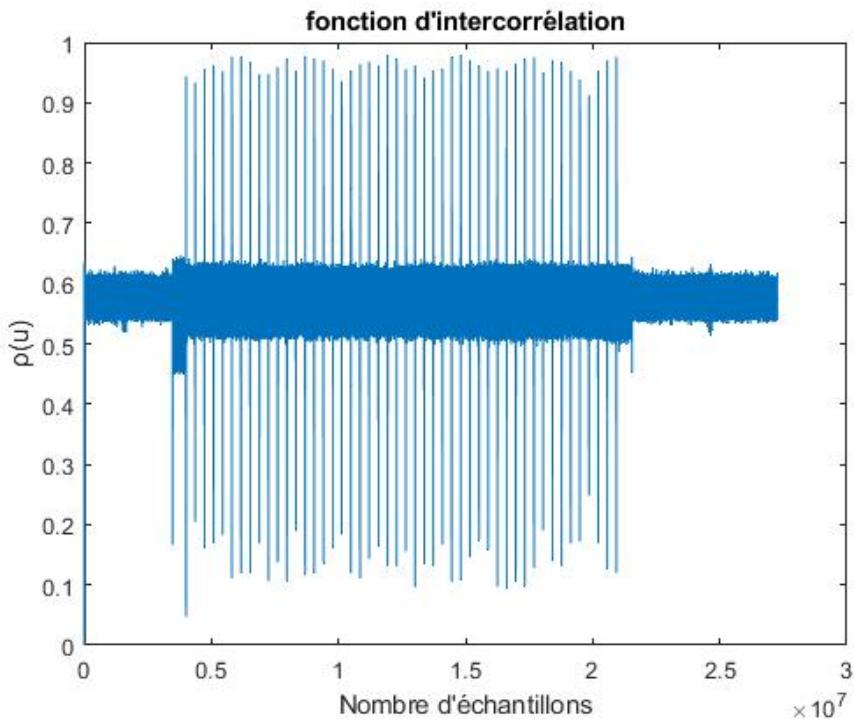


FIGURE 14 – Intercorrélation du signal reçu avec le préambule théorique, détection de 48 préambules

#### 5.4.2 Synchronisation fréquentielle

Les modules SDR HackRF One introduisent un décalage fréquentiel  $\Delta f_{Tx} + \Delta f_{Rx}$ . La synchronisation fréquentielle grossière permet d'estimer approximativement ce décalage, mais suffisamment bien pour pouvoir appliquer le filtrage adapté sans perdre une partie du signal. Les symboles différentiels étant des racines huitièmes de l'unité, on pourrait facilement estimer le décalage en détectant le maximum de la transformée de Fourier du signal reçu à la puissance huit. En effet, en notant  $\Delta f$  ce décalage, le signal élevé en puissance prend la forme :

$$\exp j(8\phi(t)\frac{\pi}{4} + 2\pi 8\Delta f t) = \exp j2\pi\phi(t) \exp j2\pi 8\Delta f t = \exp j2\pi 8\Delta f t \quad (7)$$

où  $\phi(t) \in 1, 2, 3, 4, 5, 6, 7, 8$

Dans le cas idéal ( $\Delta f$  est constant au cours du temps) la transformée de Fourier du signal de l'équation 7 est un Dirac. Ainsi, détecter son maximum revient à estimer  $8\Delta f$ .

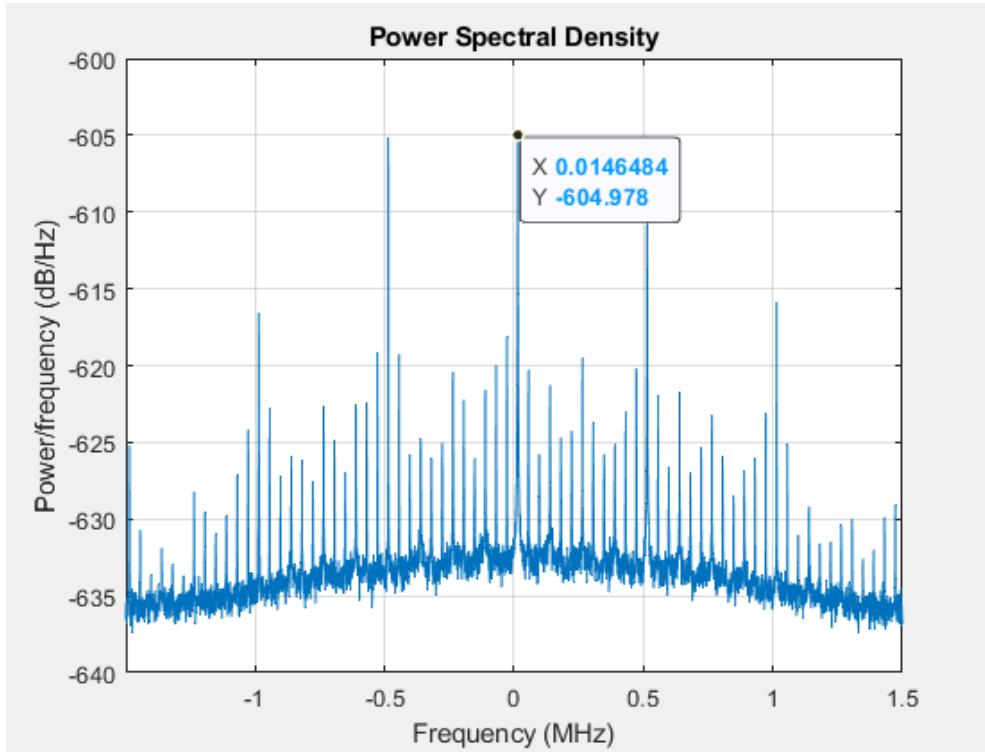


FIGURE 15 – Transformée de Fourier du signal reçu élevé à la puissance huit par méthode de Welch,  $\Delta f = \frac{1.4648e+04}{8}$

Une fois  $\Delta f$  estimé et corrigé, le filtrage adapté peut être réalisé. Cependant, cette estimation est faite à un déphasage constant près. Ainsi, après filtrage, et en se basant sur le codage différentiel on effectue une correction de phase. Le premier symbole différentiel étant connu (fixé à  $\exp(j\frac{\pi}{4})$ ), le déphasage résiduelle est estimé en repérant le déphasage sur ce premier symbole.

Après les algorithmes de synchronisation, on passe au démodulateur numérique pour récupérer notre flux binaire ; on vérifie les octets de vérifications hex AA ; on les supprime et on récupère notre image.

## 6 Simulations

D'abord on simule notre chaîne de communication sans bruit et sans canal de propagation, puis on vérifie bien que les constellations sont correctes, ensuite on ajoute un bruit gaussien et un canal qui engendre un décalage temporel et fréquentiel et on essaie de transmettre plusieurs sous paquets de l'image afin de vérifier le bon déroulement des différentes synchronisations.

## 7 Transmissions réelles d'une image avec les HackRFs (MATLAB)

Une fois les simulations validées, des transmissions réelles de l'image "background.jpg" de taille 768\*512 sont effectuées. Au fur et à mesure quand on augmente le nombre de préambules la qualité des images reçues s'avèrent meilleure. Un nombre de préambule fixé à 64 était en fin

de compte retenu. On se sert donc de notre script MATLAB d'émission pour préparer un fichier ".raw" contenant notre signal qui sera envoyé avec le HackRF One avec le gain du VGA à 32 dB et nous le recevons avec le deuxième SDR avec le gain du LNA à 24 dB. le nouveau fichier ".raw" reçu est traité avec le script MATLAB de réception qui sert à démoduler les symboles reçus et les remettre sous forme d'une image. Voici quelques résultats des tests de transmissions réels :



FIGURE 16 – Image "background.jpeg" avant transmission



(a) Image reçue en utilisant 32 préambules

(b) Image reçue en utilisant 64 préambules

FIGURE 17

Les lignes de commande utilisées pour faire l'émission du fichier ".raw" et sa réception sont les suivantes :

**Emission** : `hackrf_transfer -t /Users/stage/Desktop/fichiers_Aasma/QPSK_Tx.raw -f 1278000000 -s 3000000 -x 32 -p 1 -a 1 -d 000000000000000088869dc242e9d1b`  
**Reception** : `hackrf_transfer -r /Users/stage/Desktop/fichiers/QPSK_Rx.raw -f 1278000000 -s 3000000 -l 24 -g 24 -d 000000000000000088869dc242e9d1b`

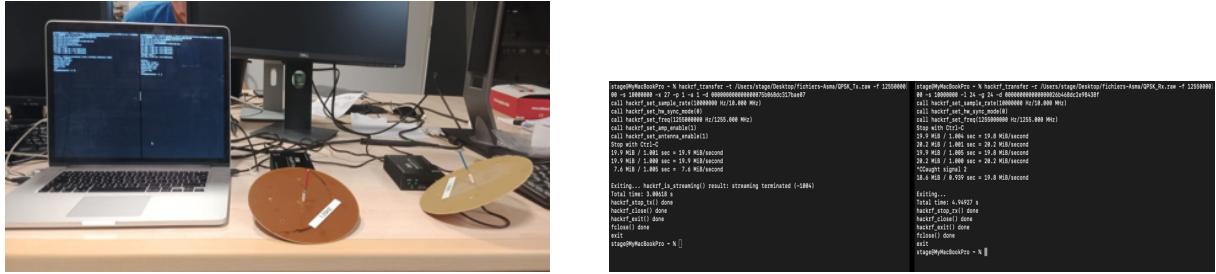


FIGURE 18 – Transmissions réelles des images

voici les paramètres utilisés dans les lignes de commandes et leurs utilités :

- **-t** Transmettre les données qui se trouve dans un fichier "raw". On place le chemin du fichier qu'on veut transmettre après le "-t".
- **-r** La réception des données transmises suivi du chemin où on souhaite stocker le nouveau fichier de la réception.
- **-f** représente la fréquence de la porteuse ; dans mon stage les deux fréquences utilisées sont 1,255 GHz et 1,278 GHz
- **-s** représente le sample rate.
- **-x** TX VGA gain, c'est le seul gain de la transmission et sa valeur peut varier entre 0 et 47dB.
- **-p** Alimentation du port d'antenne, activé s'il était égale à 1 et désactivé à 0.
- **-a** Amplificateur RF RX / TX activé si a = 1 et désactivé si a = 0.
- **-l** RX LNA gain, c'est un gain de réception qu'on peut régler selon nos besoin de la transmission et qui peut aller de 0 jusqu'à 40 dB avec un pas de 8 dB.
- **-g** RX VGA (bande de base) gain, c'est un gain de réception allant de 0 jusqu'à 62 dB et il ne faut pas que sa valeur soit très loin de la valeur du RX LNA.
- **-d** représente le serial du HackRf qu'il faut utiliser

## 8 Implémentation en C++ du bloc d'émission sur la Raspberry Pi

### 8.1 Prise et stockage des images

Pour capturer et enregistrer les images prises par la caméra USB, on utilise la bibliothèque libre OpenCV(Open Computer Vision) qui met à disposition de nombreuses fonctionnalités très diversifiées permettant de traiter les images en temps réel.

Dans un premier temps, nous avons envisagé de placer deux caméras USB différentes dans la nacelle, la première pour capturer les images du sol et la deuxième celles de l'horizon. Le code de la figure 19 permet d'alterner entre les deux caméras. Nous avons opté pour stocker des images de résolution 1920\*1080 que l'on diminue par la suite à l'étape de transmission pour des raisons d'optimisation du code d'émission.

```

#include "opencv2/opencv.hpp"
#include <iostream>
#include <unistd.h>
#include <string>
#include <cstdlib>

using namespace std;

int main(int, char**) {

    int num = atoi(getenv("NUM"));
    // open the webcams plugged in the Raspberry
    cv::VideoCapture camera(0);
    cv::VideoCapture camera2(2);
    if (!camera.isOpened()) {
        std::cerr << "ERROR: Could not open camera" << std::endl;
        return 1;
    }
    camera.set(CV_CAP_PROP_FRAME_WIDTH,1920);
    camera.set(CV_CAP_PROP_FRAME_HEIGHT,1080);
    camera2.set(CV_CAP_PROP_FRAME_WIDTH,1920);
    camera2.set(CV_CAP_PROP_FRAME_HEIGHT,1080);

    // this will contain the image from the webcam
    cv::Mat frame;
    if (num % 2 ==0)
        camera >> frame;
    else
        camera2 >> frame;
    if(frame.empty())
    {
        std::cerr << "Something is wrong with the webcam, could not get frame." << std::endl;
    }
    // Save the frame into a file
    imwrite(getenv("IMAGE"), frame); // A JPG FILE IS BEING SAVED
    //cout << "done" << endl;

    return 0;
}

```

FIGURE 19 – Code de capture d'image en C++

On utilise des variables d'environnements afin de stocker le numéro des images capturées ainsi que leurs noms et ce pour pouvoir récupérer ces informations dans les code de capture et d'émission. En effet, les variables d'environnement servent à transmettre des informations dans les processus qui se déclenchent depuis le shell, et leurs valeurs sont récupérés grâce à la fonction `getenv()`.

Dans un deuxième temps, nous nous sommes contentés d'une seule caméra pour respecter les contraintes de masse.

## 8.2 Ecoute des signaux perturbateurs

Le processus chargé de l'écoute est un processus fils obtenu grâce à l'appel système `fork()` de C++. Une fois la seconde de l'écoute écoulée, le processus père reprend la main pour lui mettre fin à l'aide de la fonction `kill()`.

L'utilisation des processus affiliés a été conçue dans le but d'éviter d'arrêter manuellement le processus d'écoute. En fonction de l'énergie du signal repéré à l'écoute, on décide si notre transmission aura lieu. En effet cette énergie donnée en équation 8 doit être inférieure à un seuil donné pour émettre.

$$E = \frac{1}{N} \sum_{k=0}^{N-1} |X(k)|^2 \quad (8)$$

où N est le nombre d'échantillons du signal d'écoute échantillonné à la fréquence  $f_e = 1MHz$ . Ce choix de la fréquence d'échantillonage s'explique par le fait que la bande du signal à émettre est égale à 1MHz, ainsi nous utilisons la totalité des échantillons du signal d'écoute pour le calcul d'énergie.

### 8.3 Contrôle de l'étage d'amplification avec la raspberry pi

Dans le but d'économiser de la batterie et d'inverser le switch, il nous a fallu mettre en place une carte alimentant sur commande le PA (Power Amplifier) et le driver ainsi qu'une sortie logique 0-1 ou 1-0 pour le switch. Cette carte est commandé par une tension de 3,3V venante de la raspberry pi dans le cas de transmission et de 0V dans le cas d'écoute. Pour cela on utilise la librairie WiringPi pour gérer les entrées/sorties du Raspberry Pi, cette dernière utilise sa propre numérotation des pins GPIO :

Raspberry Pi GPIO Header							
BCM	WiringPi	Name	Physical	Name	WiringPi	BCM	
		3.3v	1	2	5v		
2	8	SDA.1	3	4	5V		
3	9	SCL.1	5	6	0v		
4	7	1-Wire	7	8	TxD	15	14
		0v	9	10	RxD	16	15
17	0	GPIO. 0	11	12	GPIO. 1	1	18
27	2	GPIO. 2	13	14	0v		
22	3	GPIO. 3	15	16	GPIO. 4	4	23
		3.3v	17	18	GPIO. 5	5	24
10	12	MOSI	19	20	0v		
9	13	MISO	21	22	GPIO. 6	6	25
11	14	SCLK	23	24	CEO	10	8
		0v	25	26	CE1	11	7
0	30	SDA.0	27	28	SCL.0	31	1
5	21	GPIO.21	29	30	0v		
6	22	GPIO.22	31	32	GPIO.26	26	12
13	23	GPIO.23	33	34	0v		
19	24	GPIO.24	35	36	GPIO.27	27	16
26	25	GPIO.25	37	38	GPIO.28	28	20
		0v	39	40	GPIO.29	29	21
BCM	WiringPi	Name	Physical	Name	WiringPi	BCM	

FIGURE 20 – Numérotation des pins GPIO de la carte Raspberry Pi

Pour générer la tension de commande on choisit la pin 40 qui correspond au GPIO.29 et grâce aux fonctions digitalWrite(Pin,HIGH) et digitalWrite(Pin,LOW) on peut générer 3,3V ou 0V.

## 8.4 Emission en code Morse de l'identifiant du club radio en C++ du bloc d'émission sur la Raspberry Pi

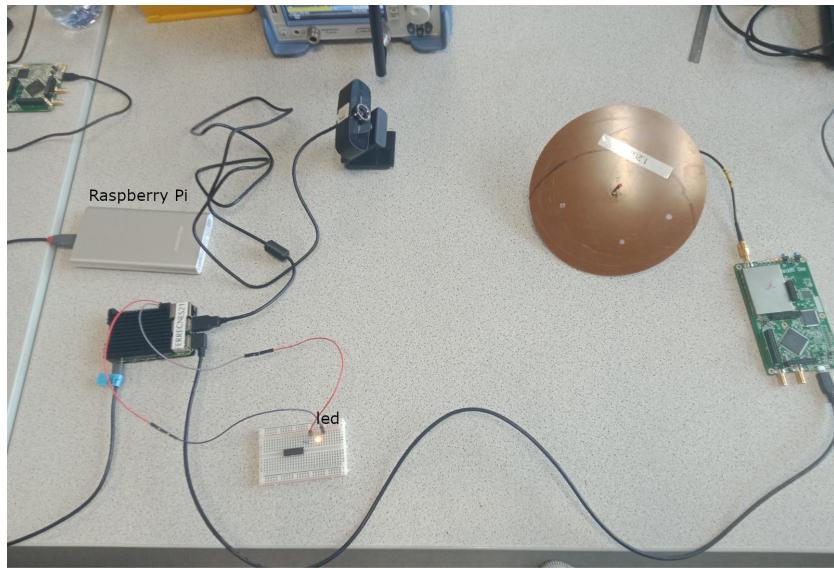


FIGURE 21 – Vérification de la tension de commande générée par la raspberry pi à l'aide d'un led

## 8.4 Emission en code Morse de l'identifiant du club radio

On code l'identifiant du club radio F6KQH sous forme d'un vecteur de 0 et 1, qui sera stocké dans un fichier ".raw" et envoyé avec le HackRF One avant la transmission d'une image.

## 8.5 Emission de l'image

L'émetteur est implémenté en C++ suivant la même logique du script MATLAB de l'émission. On peut schématiser la chaîne d'émission pour le code C++ comme ceci :

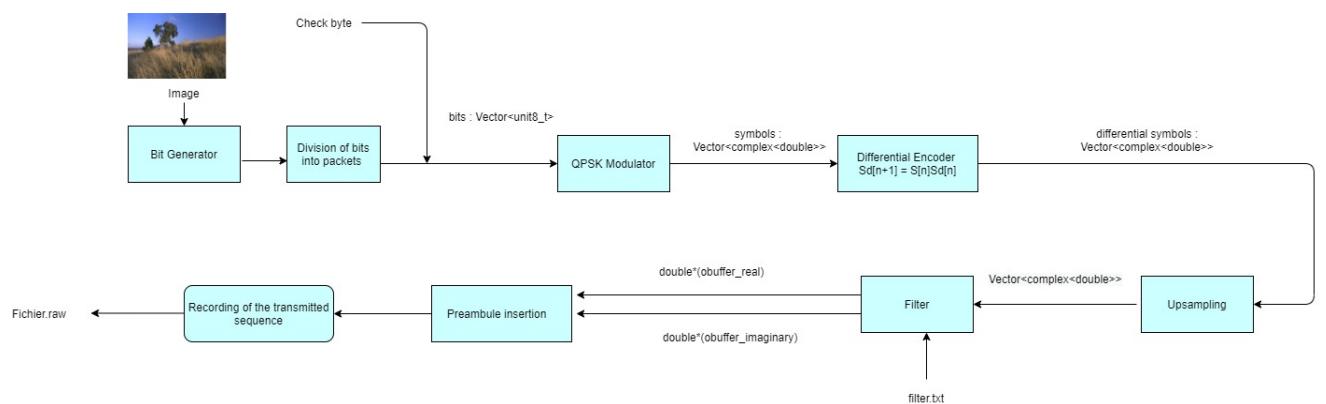


FIGURE 22 – Chaîne d'émission implémentée en C++

où chaque bloc représente une classe en C++. Pour le passage dans le filtre de mise en forme, il faut enregistrer la matrice de ce filtre accessible sur Matlab en un fichier texte et faire la convolution à la main à partir de boucles for. La première solution d'implémenter la convolution est un calcul de base comme le montre la fonction suivante :

```

int convolve_naive(float* in, float* out, int length,
                   float* kernel, int kernel_length)
{
    for(int i=0; i<=length-kernel_length; i++){
        out[i] = 0.0;
        for(int k=0; k<kernel_length; k++){
            out[i] += in[i+k] * kernel[kernel_length - k - 1];
        }
    }
    return 0;
}

```

FIGURE 23 – Calcul de convolution, méthode de base

Cette solution prend beaucoup de temps. la deuxième solution retenue consiste à réimplémenter cela avec des instructions SIMD. Cela se base sur la vectorisation de l'algorithme pour calculer 4 échantillons de sortie en parallèle. En effet, si le filtre est de longueur N, alors nous créons N 4-vecteurs. Nous copions ensuite chaque élément du filtre dans chaque élément du 4-vecteur correspondant. Nous pouvons alors calculer 4 valeurs de sortie en parallèle.(Le code est en annexe )

## 9 Mesures de puissance

Afin de s'assurer que la puissance du signal est suffisamment importante pour recevoir un signal de bonne qualité, des mesures de puissances ont été réalisées à l'aide d'un analyseur de spectre. Les puissances étaient mesurées en fonction du gain VGA de la radiogiciel de l'émission. On a obtenu une puissance maximale d'émission de 25,5 dBm, comme le montre les figures suivantes :

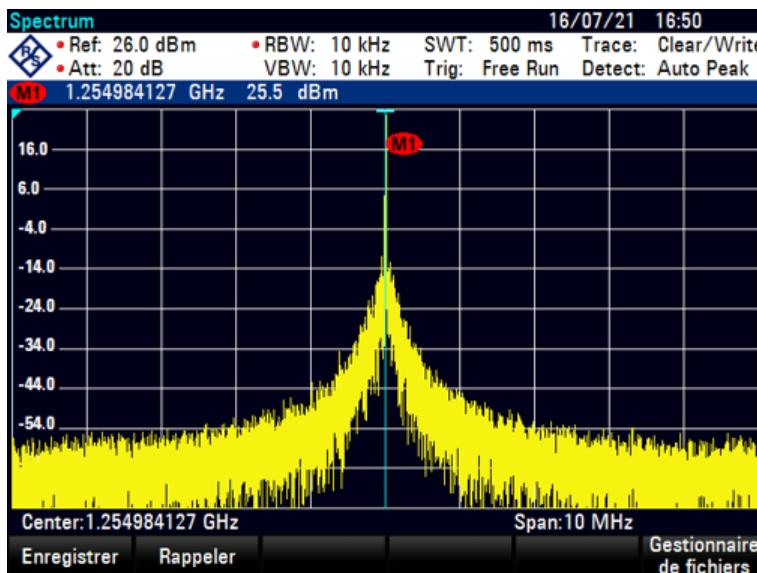


FIGURE 25 – Calcul de convolution, méthode de base

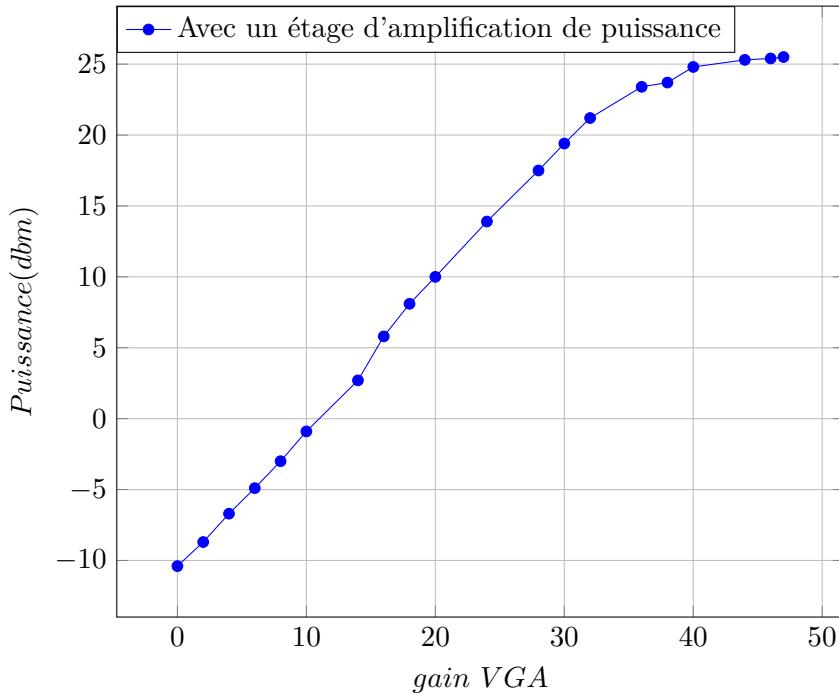


FIGURE 24 – Puissance d'une onde entretenue (continuous wave) d'amplitude maximale 127 en fonction du gain vga de la hackrf.

## 10 Tests en conditions de vol

### 10.1 Tests à vide

A 30 Km d'altitude, la pression est basse comparée à celle sur le sol. C'est pourquoi il est indispensable de vérifier son impact sur le matériel utilisé. En effet, en l'absence de l'air les composants électroniques peuvent subir un réchauffement. Afin de vérifier l'endurance du système, des tests à vide ont été réalisés au fablab en utilisant une cloche à vide. Lors d'un premier test de durée d'une heure, on était témoin d'un réchauffement de la carte raspberry pi utilisée, ce qui a limité son fonctionnement. La solution qui a été envisagée est de monter un radiateur sur la carte pour dissiper la température. Celle ci s'est avérée efficace, étant donné qu'un test de presque 3 heures a été validé.



FIGURE 26 – Tests à vide

## 10.2 Complexités d'alimentation du module

Nous avions dû faire face à plusieurs difficultés lors de l'alimentation du module d'émission, que l'on a essayé de résoudre en toute urgence deux semaines avant le lancement du ballon. En effet, au début on utilisait des piles lithium avec un simple montage capacités - régulateur linéaire 9V - 5V, cependant, le courant demandé était si important pour le composant que la chaleur qu'il dégageait par effet joule était bien au-dessus de sa plage de fonctionnement et se détruisait. Une deuxième idée était d'utiliser un régulateur buck - alimentation à découpage, cependant le courant n'était toujours pas assez important et la température était très élevée, et sera d'autant plus durant le vol en l'absence d'air. On a envisagé comme pour la raspberry d'utiliser un radiateur pour des fins de refroidissement. Pour ce faire, il fallait calculer la dimension exacte de ce dernier qui répondra aux lois thermiques de rayonnement du corps noir en appliquant la formule suivante :

$$S = \frac{P}{T^4 \sigma \epsilon} \quad (9)$$

où  $\sigma$  la constante de Boltzmann en  $W.m^{-2}.K^{-4}$  et  $\epsilon$  le coefficient d'émissivité de l'aluminium. Le résultat obtenu est  $500cm^2$ . un radiateur d'une telle dimension ne peut évidemment pas être intégré dans la nacelle.

En fin de compte, on a opté pour l'utilisation d'une batterie lithium polymère (Li-Po) capable d'apporter une alimentation de 5V continue et stable (régulateur interne) et de fournir 3A.



FIGURE 27 – Batterie lithium polymère (Li-Po)

**Remarque :**

Il faut être vigilant lors de la manipulation des piles lithium. En effet, celles-ci sont très inflammables en cas de court-circuit ou déchargement important. Ceci aurait pu nous arriver dans le cas où le ballon n'est pas retrouvé rapidement : le déchargement complet des piles provoque, dans de basses températures, un court-circuit interne qui répand un liquide comburant et carburant à la fois. De plus, les piles n'ayant pas exactement une tension égale, les mettre en parallèle impliquerait que certaines alimenteraient d'autres ce qui ferait rentrer du courant susceptible de faire exploser la pile.

### 10.3 Tests en basses températures

Lors du vol, le ballon sera soumis à des températures pouvant atteindre -15°C à l'extérieur. Il faut donc s'assurer que les alimentations sont capables de fournir une tension et un courant suffisants pour de telles températures. Des tests dépassant les trois heures ont été réalisés avec l'utilisation de la batterie précédemment mentionnée, ceux-ci étaient validés pour des températures allant de la température ambiante jusqu'à -10° C. En dessous de -10°C, des coupures d'alimentation ont été observées. Ainsi, nous avons programmé la raspberry de manière à ce qu'elle relance le code d'émission immédiatement après son redémarrage.



FIGURE 28 – Tests en basses températures (-15°C)

## 11 Tests de transmission sur longues distances

Une fois tous les tests en conditions du vol et les mesures de puissance réalisés et validés, nous avons essayé de faire des transmissions sur longues distances :

- Des tests de réception avec l'antenne de l'école et la nacelle positionnée dans le parking puis dans un des bâtiments de l'université de bordeaux (400 m) ont été validés, mais en ajoutant un amplificateur de puissance dans la chaîne de réception au club radio.
- Des tests de transmissions depuis Bouliac qui n'étaient pas validés dû à l'incapacité de détecter le signal avec le code de réception.

### Pistes d'amélioration :

- Utilisation d'un filtre sélectif pour la réception afin de bien détecter le signal et éliminer le bruit.
- Faire un bilan de liaison de la hackRF de reception pour déterminer sa sensibilité, et étudier la possibilité de recevoir le signal avec une USRP.

## 12 Conclusion

Pour conclure, je tiens à ajouter que ce stage d'application effectué dans le cadre de ma deuxième année d'école d'ingénieurs était très enrichissant.

En effet, durant cette période et grâce aux ressources et aux explications de mes encadrants, j'ai pu mettre en application mes connaissances en communications numériques et traitement d'images. De plus, ce stage m'a permis de sortir du cadre scolaire et d'apprendre à travailler en

équipe. Par ailleurs, c'était aussi une occasion pour moi de monter en compétence et de gagner en expérience car durant cette période j'ai découvert et exploité de nouveaux matériaux lié au domaine de électronique. Ce qui m'a permis de prendre connaissance de mes points forts et mes points faibles en tant qu'étudiante ingénierie. Ceci va certainement m'aider à évoluer et à réussir mon projet professionnel.

## 13 Annexes

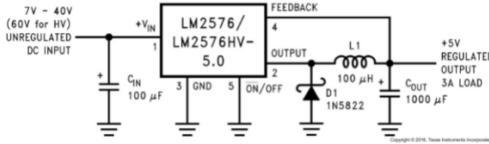


FIGURE 29 – Circuit électrique de puissance (buck) relatif à la carte RSPB

```
float* Filter::convolve(float* in, float* kernel, int input_length, int kernel_length)[
    int nconv = input_length+kernel_length-1;
    (*(&len_conv)) = nconv;
    float *out = (float*) calloc(nconv, sizeof(float));

    float* in_padded = (float*)(alloca(sizeof(float) * (input_length + 8)));
    float32x4_t* kernel_many = (float32x4_t*)(alloca(16 * kernel_length));
    float32x4_t block;

    float32x4_t prod;
    float32x4_t acc;

    // surrounding zeroes, before and after
    vst1q_f32(in_padded, vdupq_n_f32(0));
    memcpy(&in_padded[4], in, sizeof(float) * input_length);
    vsti1q_f32(in_padded + input_length + 4, vdupq_n_f32(0));

    // Repeat each kernel value across a 4-vector
    int i;
    for (i = 0; i < kernel_length; i++) {
        kernel_many[i] = vdupq_n_f32(kernel[i]); // broadcast
    }
    for (i = 0; i < input_length + kernel_length - 4; i += 4) {

        // Zero the accumulator
        acc = vdupq_n_f32(0);

        int startk = i > (input_length - 1) ? i - (input_length - 1) : 0;
        int endk = (i + 3) < kernel_length ? (i + 3) : kernel_length - 1;
        /* After this loop, we have computed 4 output samples
         * for the price of one.
         */
        for (int k = startk; k <= endk; k++) {

            // Load 4-float data block. These needs to be an unaligned
            // load (_mm_loadu_ps) as we step one sample at a time.
            block = vld1q_f32(in_padded + 4 + i - k);
            prod = vmlaq_f32(block, kernel_many[k]);

            // Accumulate the 4 parallel values
            acc = vaddq_f32(acc, prod);
        }
        vst1q_f32(out + i, acc);
    }
    // Left-overs
    for (; i < input_length + kernel_length - 1; i++) {

        out[i] = 0.0;
        int startk = i >= input_length ? i - input_length + 1 : 0;
        int endk = i < kernel_length ? i : kernel_length - 1;
        for (int k = startk; k <= endk; k++) {
            out[i] += in[i - k] * kernel[k];
        }
    }
}

return out;
]
```

FIGURE 30 – Code de convolution optimisé avec des instructions SIMD



FIGURE 31 – Préparation de la nacelle avant le vol

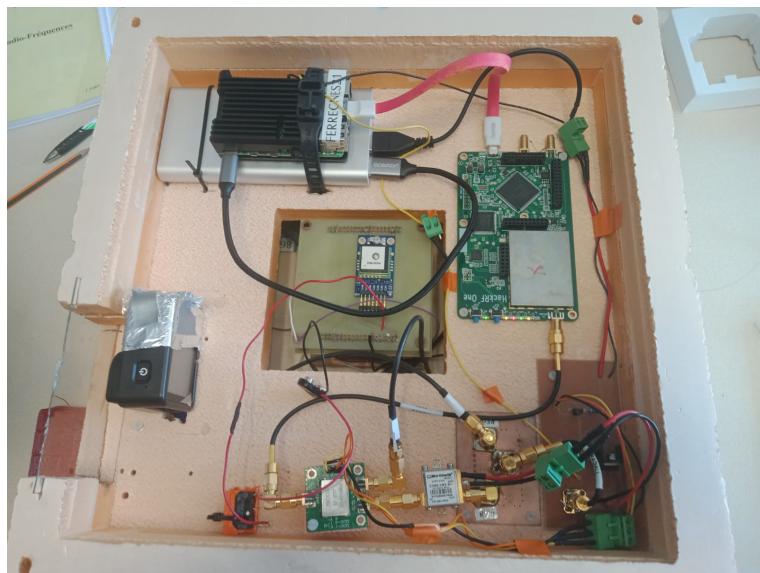


FIGURE 32 – Module de transmission d'images embarqué sur la nacelle



FIGURE 33 – Image prise par la caméra GoPro



Cette année, plusieurs élèves de différentes filières de l'ENSEIRB-MATMECA, promotion 2019, ont réalisé un projet afin de célébrer les 100 ans de notre école d'ingénieur. Encadré par plusieurs enseignants et bénéficiant du partenariat du CNES et de Planète Science, celui-ci vise à lâcher un ballon sonde contenant plusieurs expériences embarquées à son bord.



Plusieurs objectifs ont été définis : capture et traitement d'images, analyses de données météorologiques, adaptation de l'électronique de puissance, programmation des systèmes embarqués pour traitement de données, stockage et modulation du signal pour transmissions des données par ondes radios, traitement en temps réel des informations reçues au sol et animation en live de la journée du vol.

Au cours de ce projet, l'équipe a dû développer des compétences allant au-delà de leur filière d'étude comme l'utilisation de machines industrielles utilisées après un travail de C.A.O., une organisation de travail importante liée à un cahier des charges et une législation très stricts ainsi qu'une communication extérieure vitale.

Soyez nombreux à regarder le live youtube le jour j :

[https://www.youtube.com/embed/live\\_stream?channel=UCRL5Oyl2xB2LGjnnmzF19MQ](https://www.youtube.com/embed/live_stream?channel=UCRL5Oyl2xB2LGjnnmzF19MQ)

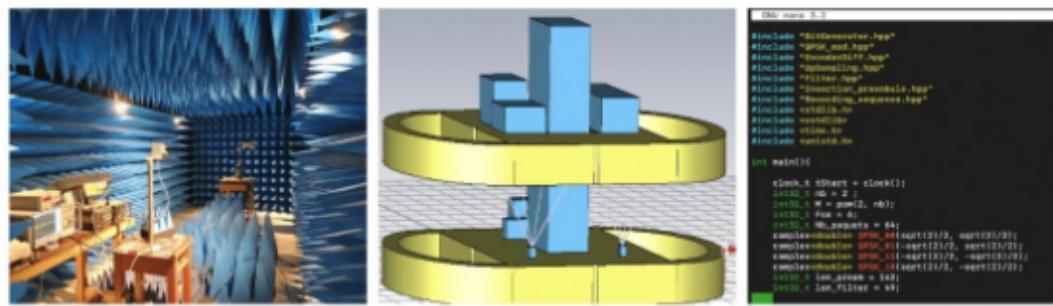


FIGURE 34 – flyer de présentation du projet pour l'école