À l'écoute des messages transmis par satellite en orbite basse : Iridium

Jean-Michel Friedt, 17 mai 2024

Nous explorons les signaux de communication, voix et messages aéronautiques ACARS ainsi que messages textuels de type SMS, et de localisation de la constellation de satellites en orbite basse Iridium, reçus par radio logicielle connectée à une. antenne GPS adaptée pour cette utilisation, afin de décrire la séquence d'utilisation de `gr-iridium` et `iridium-toolkit`

1 Introduction

Avant l'âge de l'espace, la communication par liaison radiofréquence se limitait à la portée à vue dans le mode de propagation le plus simple, ou imposait de communiquer à basse fréquence (qualifié en radio de "haute fréquence", HF compris entre 300 kHz et 30 MHz) pour bénéficier du guide d'onde que forme l'ionosphère avec la surface de la planète. Quelques modes de propagation exotiques (*tropospheric scatter*) permettaient en très haute fréquence (VHF entre 30 et 300 MHz) des liaisons au-delà de l'horizon, mais au détriment d'une communication incertaine et nécessitant des antennes immenses. Ces installations ont été utilisées jusqu'aux premiers satellites pour communiquer avec les régions lointaines d'Alaska, de Sibérie ou les îles au milieu de la Méditerranée par exemple.

Le lancement en 1962 de Telstar1 à 1000 km d'altitude a permis le premier lien transatlantique relayé depuis l'espace, avec toutes les évolutions que nous connaissons aujourd'hui des satellites géostationnaires vers les essaims mais surtout avec Iridium que nous allons étudier en détail.

2 La communication par satellites

En plaçant un satellite à 36000 km de la surface de la Terre pour qu'il tourne à la même vitesse que la rotation de la Terre (orbite géostationnaire), un satellite voit presque la moitié du globe et permet donc une liaison sur une bien plus grande distance que le permettrait une liaison entre deux émetteurs situés sur la surface du globe. De la géométrie élémentaire (Fig. 1) indique que l'angle ϑ entre l'horizon HH' pour un satellite à une altitude h et un point P sur la surface du globe de rayon R est donné par $\cos(\vartheta) = \frac{R}{R+h}$ et le rayon du cercle vu depuis le satellite est $R\vartheta = R\text{acos}\left(\frac{R}{R+h}\right)$. Compte tenu du rayon de la Terre de 6400 km, un satellite géostationnaire permet à deux interlocuteurs distants de 17820 km (2×8910 km) de communiquer au travers de ce relais. Ce mode de communication dans lequel un satellite reçoit un message émis depuis le sol et le relaie vers un récepteur est nommé *bent pipe* [1], dans lequel un transpondeur reçoit un signal sur une fréquence montante – par exemple 14 GHz dans le cas de Telstar11N [2] – et retransmet tout ce qu'il reçoit en se contentant d'amplifier et transposer en fréquence vers la porteuse descendante – par exemple 11 GHz pour Telstar11N. Divers canaux sont loués pour diverses utilisations, qu'il s'agisse de communiquer de la voix, des données ou de la télévision par exemple.

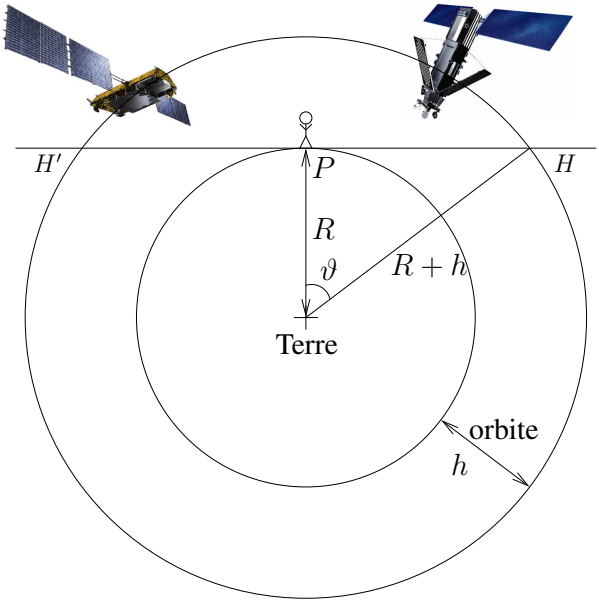


Figure 1: Portée de la liaison d'un satellite survolant à altitude h la Terre de rayon R. À droite un satellite Iridium original avec ses immenses panneaux d'antennes réfléchissant le soleil lors des *iridium flares*, et à gauche les nouveaux Iridium Next de Thalès Alenia Space.

Pour revenir au cas de Telstar1 à 1000 km d'altitude, le rayon de la zone vue depuis l'espace est de 3365 km donc suffisant pour couvrir les 5000 km entre la Bretagne et le Maine sur la côte Est américaine.

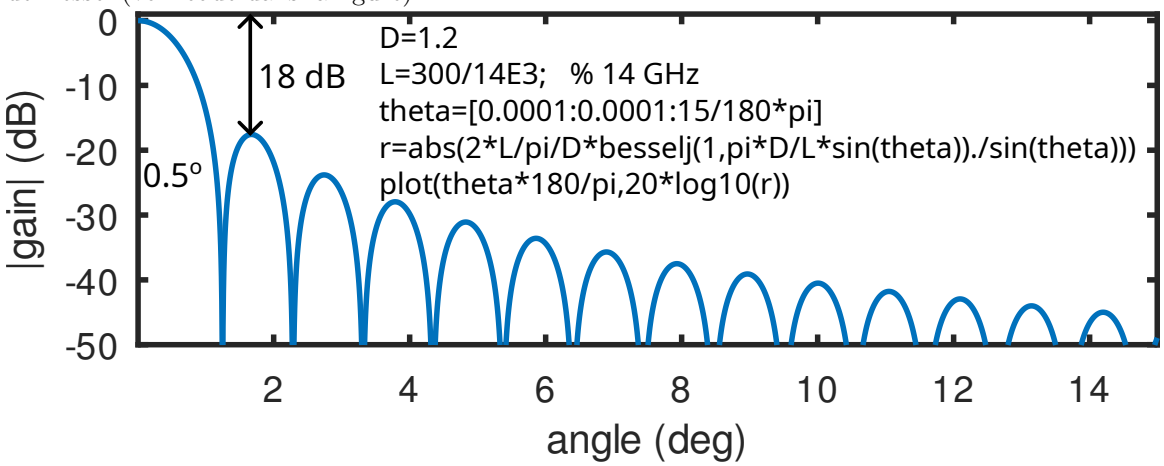
Cependant, ce mode de communication a un gros défaut : alors que l'émetteur au sol (nommé VSAT pour *Very Small Aperture Terminal*) essaie de ne viser que le satellite relais, en pratique il illumine

toujours un peu les voisins. Or un observateur capable de recevoir le signal relayé *et la retransmission des voisins* peut retrouver, par trilatération, l'emplacement de l'émetteur. Ce n'est pas facile car la puissance reçue et re-émise par les satellites voisins de celui visé est faible, mais avec des oreilles suffisamment grandes cela est possible (Fig. 2). Ce service est même proposé commercialement [4] pour identifier l'emplacement d'interférants, volontaires [5] ou non, de liaisons satellites par relais géostationnaire.

FIGURE 2 – Des grandes oreilles vues fortuitement au détour d'une randonnée en vallée de Chevreuse sur le GR11.

Diagramme de rayonnement et lobes latéraux d'une parabole

Il peut être intéressant de noter que l'excellente performance d'un réflecteur parabolique tient justement aux faibles lobes latéraux qui évite de perdre de la puissance en dehors de la direction principale de visée. Cependant un réflecteur de dimensions finies a nécessairement des lobes latéraux : on montre que le diagramme de rayonnement dans l'axe de révolution du paraboloïde est une fonction de Bessel (voir code dans la figure).



Les 0,5° indiquent la largeur de faisceau à -3 dB, tandis que le premier null se trouve à 1,5 °. Ce calcul pour une parabole de $D = 1,2$ m à 11 GHz est représentatif de l'antenne qui pointe depuis Telstar11N vers l'Europe[3], et 1,5° depuis une altitude de 36000 km (en pratique 39000 km en tenant compte de la position du satellite au dessus de l'Atlantique par $37,5$°W) couvrent un cercle d'environ 1000 km de rayon soit l'Europe. Ce calcul met surtout en évidence la difficulté à écouter le signal rayonné vers l'Europe en dehors de ce continent, avec plus de 18 dB d'atténuation dans les lobes suivants.

Ce risque de localiser l'origine de l'émetteur explique l'intérêt des utilisateurs soucieux de leur anonymité – en premier lieu les militaires – de rechercher une alternative.

3 La constellation Iridium

Dans les années 1990, avant le déploiement massif des relais de téléphonie mobile à la surface du globe, deux philosophies se concurrençaient : la téléphonie cellulaire avec ses relais au sol, et l'utilisation de relais dans l'espace. Cependant communiquer avec un satellite à 36000 km nécessite un faisceau relativement directif (pour ne pas trop illuminer les satellites voisins) et puissant pour que le signal

reçu soit suffisamment clair pour être amplifié et retransmis. Peu d'utilisateurs de téléphone mobile désirent pointer une antenne de quelques décimètres à une paire de mètres de diamètre vers un satellite géostationnaire pour effectuer un appel. Motorola s'est donc lancé dans la construction d'une constellation de satellites en orbite basse capable de relayer les signaux émis depuis le sol. Les satellites en orbite basse (LEO) volent à une altitude autour de 800 km, et Iridium n'y fait pas exception : 413 miles nautiques [6] font 765 km, et la même considération que précédemment indique que chaque satellite voit au sol un disque de 3000 km de rayon. Que faire donc si deux interlocuteurs sont distants de plus de 6000 km ou si aucune station au sol n'est dans ce rayon pour réceptionner le signal ? L'innovation d'Iridium tient au fait que *les satellites parlent entre eux* et relaient les informations de satellite en satellite par un lien micro-ondes avant de transmettre vers une des stations au sol de la société. Ces relais multiples anonymisent la source de l'appel et ceci explique que lorsque Morotola abandonne son soutien financier à Iridium qui est alors sur le point de faire faillite et de désorbiter sa constellation, le Pentagone sauve la société et ses satellites en s'engageant à acheter suffisamment de services pour maintenir la viabilité financière [1].

Même si les satellites ne sont plus visibles dans le ciel comme ils l'étaient lorsque leurs immenses réseaux d'antennes s'alignaient convenablement avec le soleil couchant (*iridium flare*), ils continuent à retransmettre les signaux. Comme toute technologie datant de quelques décennies, la réception de ces signaux est maintenant accessible à tout amateur éclairé possédant une radio logicielle puisque l'outil de décodage `gr-iridium` a été rédigé par deux hackers (au sens noble du terme) allemands du CCC de Munich, Sec et Schneider [7, 8].

4 Utilisation de `gr-iridium`

Comme tout `gr-*`, `gr-iridium`, disponible à `https://github.com/muccc/gr-iridium.git`, s'appuie sur GNU Radio pour acquérir et traiter les données obtenues par un récepteur de radio logiciel. Compiler `gr-iridium` se réduit donc à cloner le dépôt git, `mkdir build && cd build && cmake ../ && make && sudo make install` sous réserve d'avoir une installation fonctionnelle de GNU Radio. Compte tenu de la complexité actuelle de compiler GNU Radio, il est probablement judicieux de s'appuyer sur sa distribution binaire favorite, par exemple avec le paquet `gnuradio-dev` sous Debian/GNU Linux. La cross-compilation avec une version de GNU Radio issue de Buildroot se fait sans trop de douleur en ajoutant, en supposant `$BR` le répertoire où Buildroot est installé, par
`cmake -DCMAKE_TOOLCHAIN_FILE=$BR/output/host/usr/share/buildroot/toolchainfile.cmake ../`
Il est pratique de rajouter l'option `-DCMAKE_INSTALL_PREFIX=/tmp/rpi` qui placera l'arborescence dans `/tmp/rpi` lors du `make install` qui se copie aisément (`scp -r`) vers le répertoire `/usr` de la cible embarquée une fois `gr-iridium` cross-compilé.

`gr-iridium` se contente de passer des signaux radiofréquences modulés en QPSK dans les diverses sous-bandes de fréquences en bits, mais nécessite par ailleurs un ensemble d'outils regroupés dans `iridium-toolkit` à `https://github.com/muccc/iridium-toolkit` qui ne sont que des scripts Python et n'impliquent donc pas de compilation. `iridium-toolkit` permettra de regrouper les bits et les phrases pour former les messages intelligibles.

Avant de pouvoir acquérir un signal, il faut une antenne adéquate. Contrairement aux satellites géostationnaires qui sont toujours vus dans la même direction depuis le sol, les satellites en orbite basse parcourent le ciel d'horizon à horizon en 15 minutes environ, et une antenne omnidirectionnelle est souhaitable. Tout comme GPS L1 et Galileo E1 qui émettent autour de 1575,42 MHz , Iridium communique en bande L mais autour de 1622 MHz. Ainsi, l'élément rayonnant et le pré-amplificateur d'une antenne GPS semblent idéaux pour Iridium, répondant aux exigences de bandes de fréquences et de diagramme de rayonnement, mais le filtre passe-bande qui justement évite de saturer le récepteur GPS par les signaux de communication empêche la réception du signal qui nous intéresse. Il faut donc commencer par retirer ledit filtre afin d'obtenir une antenne fonctionnelle, en le remplaçant par exemple par un fil. La Fig. 3 illustre la fonction de transfert d'une antenne GPS avant et après avoir retiré le filtre passe-bande.

Le lecteur moins aventureux ou bricoleur pourra acquérir auprès de `https://www.rtl-sdr.com/buy-rtl-sdr-dvb-t-dongles/` une antenne active sous la nomenclature "*RTL-SDR Blog Active L-Band 1525 – 1660 Inmarsat to Iridium Patch Antenna Set*" (60 US$) mais dans tous les cas on cherchera à utiliser une antenne omnidirectionnelle, et évitera une parabole, directive, qui n'observera qu'un angle

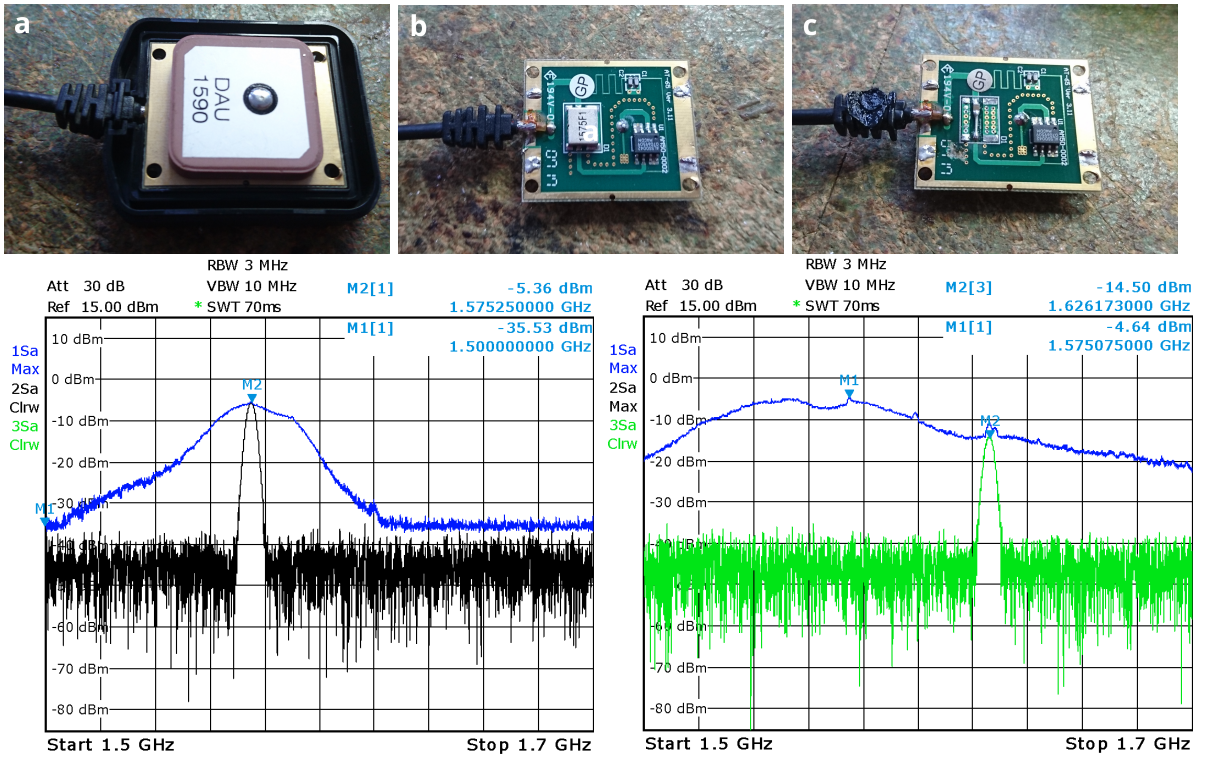


FIGURE 3 – Haut : une antenne GPS avec sa céramique métallisée (a) pour fournir une antenne compacte en bande L autour de 1600 MHz en polarisation circulaire ; la face arrière (b) avec le filtre passe-bande identifié par sa nomenclature 1575 correspondant à la partie entière de la fréquence porteuse de L1/E1 ; le filtre retiré et remplacé (c) par un fil et un condensateur en série pour couper la composante DC. Bas : fonctions de transfert de l'antenne avant (gauche) et après (droite) retrait du filtre. Les performances ne sont pas parfaites mais suffisantes à 1622 MHz.

solide infime de la sphère céleste alors que nous voulons observer les passages d'horizon à horizon des satellites.

Un unique câble coaxial relie l'antenne et la radio logicielle : afin d'alimenter le pré-amplificateur de l'antenne active, nous devons prendre soin de superposer au signal radiofréquence allant de l'antenne vers le récepteur une tension d'alimentation de 5 V. Pour séparer la contribution radiofréquence de l'alimentation DC, un T de polarisation est utilisé proche du récepteur. Ce T de polarisation peut s'obtenir commercialement, ou facilement être fabriqué au moyen d'un condensateur en série avec le récepteur (afin de couper la composante DC qui endommagerait l'amplificateur et le récepteur radiofréquence) et une inductance en série avec l'alimentation (pour éviter de perdre le signal radiofréquence dans la source de tension). À $f = 1622$ MHz, un condensateur de $C = 10$ nF présente une impédance $|Z_C| = \frac{1}{2\pi C f} = 10$ mΩ très petit devant les 50 Ω d'impédance d'entrée du récepteur radiofréquence, tandis qu'une inductance $L = 1$ μH présente une impédance $|Z_L| = 2\pi L f = 10$ kΩ très grand devant 50 Ω, évitant donc de perdre le précieux signal radiofréquence dans l'alimentation. On prendra cependant soin de sélectionner des composants de qualité micro-ondes dont les caractéristiques sont celles attendues aux fréquences de fonctionnement.

Munis de l'antenne, du T de polarisation et de divers récepteurs de radio logicielles (Fig. 4), nous sommes prêts pour nos premières acquisitions. Commençons par valider le montage et l'exposition de l'antenne au ciel en traçant le spectre dont nous mémorisons le maximum. Les performances des divers récepteurs de radio logicielle devient immédiatement visible : un récepteur RTL-SDR avec son *frontend* R820T2 et convertisseur analogique-numérique vers USB RTL2832U ne peut échantillonner que 2,4 MHz de bande passante et 1622 MHz est à la limite de la bande accessible. En pratique, compte tenu des contraintes sur la fréquence d'échantillonnage pour faciliter le décodage, seuls 2 MHz de bande passante sont disponibles, permettant de voir un signal mais de décoder bien peu de messages. Plus performante

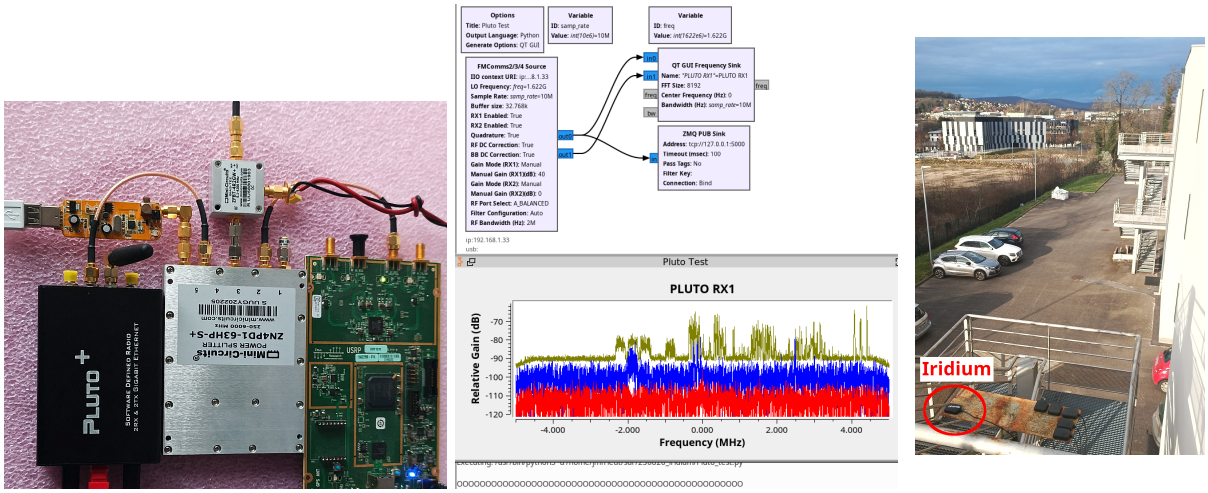


FIGURE 4 – Gauche : photo de famille de tous les récepteurs SDR connectés à l'antenne active modifiée au travers du T de polarisation (ici composant commercial MiniCircuits) au travers d'un diviseur $1 \rightarrow 4$. Au milieu, la capture d'écran du spectre acquis par Pluto+ : la séquence de O en bas du graphique (Overflow) indiquent un dépassement de capacité et perte de données. Il faudra abaisser la fréquence d'échantillonnage à 5 MHz pour ne plus perdre de données malgré la communication sur Gb Ethernet (câble rouge en bas à gauche de la photo de gauche). La B210 (droite) pourra transférer les 10 Méchantillons/s sur son bus USB3. Le récepteur RTL-SDR (en haut à gauche de la photo de gauche) est de toute façon limité à moins de 2,4 Méchantillons/s. Droite : emplacement de l'antenne de réception, en haut d'une issue de secours de l'École Nationale Supérieure de Mécanique et des Microtechniques (ENSMM) à Besançon, avec un horizon dégagé vers le nord et l'ouest mais partiellement bloqué vers le sud-est par le bâtiment aux élévations les plus basses.

mais plus chère, la Pluto+ à un peu moins de 400 euros peut aisément recevoir 1622 MHz avec son *frontend* AD9361, mais le Zynq 7010 a bien du mal à transmettre sur l'interface Gb Ethernet plus de 5 Méchantillons/seconde. Finalement l'Ettus Research B210, dont le coût est devenu prohibitif à plus de 2000 euros, permet de transmettre sur son bus USB3 plus de 10 Méchantillons/seconde, fournissant donc le plus de messages à décoder puisque couvrant la majorité des canaux (Fig. 5).

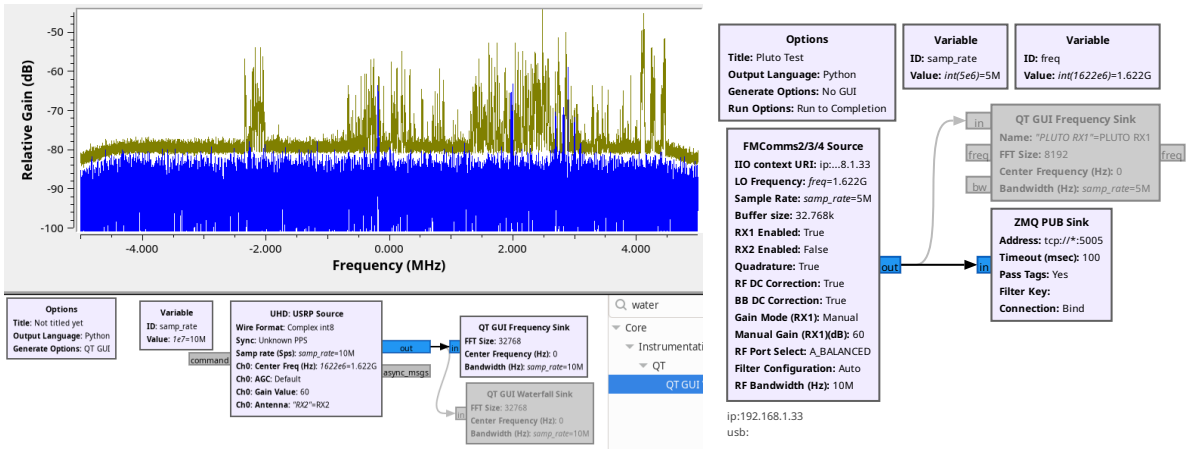


FIGURE 5 – Gauche : validation de la chaîne d'acquisition de Iridium par une plateforme de radio logicielle B210 pour en afficher le spectre pour vérifier que l'antenne est convenablement connectée et polarisée, mais à terme `gr-iridium` fournira sa propre configuration d'acquisition. Droite : contrairement à la B210, la Pluto+ n'est pas supportée par `gr-iridium` mais n'est pas exclue puisque la sortie ZeroMQ PUB pourra alimenter le décodeur Iridium qui accepte un ZeroMQ SUB comme source de données.

L'outil pour capturer les trames Iridium se nomme `iridium-extractor` dans le répertoire `apps/` de `gr-iridium` et il prend en argument le fichier de configuration et un facteur de décimation

```
iridium-extractor -D 4 --multi-frame ./examples/rtl-sdr.conf
```

pour le RTL-SDR par exemple, ou

```
iridium-extractor -D 4 --multi-frame ./examples/usrp-b2x0-uhd.conf
```

pour la B210. On notera que la Pluto+ n'est pas supportée nativement. Cependant, un ajout récent de `gr-iridium` est d'accepter un flux de données venant d'une socket ZeroMQ, signifiant que tout récepteur supporté par GNU Radio peut être utilisé en connectant le bloc source permettant l'acquisition vers un ZeroMQ Publish qui se chargera d'alimenter (Fig. 5)

```
iridium-extractor -D 4 --multi-frame ./examples/zeromq-sub.conf
```

On prendra soin dans ce cas d'utiliser bien entendu le même port (dernier argument de `address`) et surtout d'avoir la même configuration de passage du Tag (désactivé par défaut dans GNU Radio mais actif dans `gr-iridium` – sans une configuration identique la communication ne s'effectue pas). Le flux en provenance du bloc ZeroMQ Publish de GNU Radio est supposé être formé de nombres flottants codés sur 32 bits (simple précision) alternant partie réelle et imaginaire des complexes.

La sortie de `iridium-extractor` peut être soit redirigée vers un fichier pour post-traitement, soit *pipé* vers `iridium-parser.py` de `iridium-toolkit` pour un décodage en temps réel sous réserve que la puissance de calcul soit suffisante, pour donner une commande de la forme

```
iridium-extractor -D 4 --multi-frame ./examples/zeromq-sub.conf \
   | python3 -u ../iridium-toolkit/iridium-parser.py -o zmq
```

Supposons dans un premier temps que nous redirigeons la sortie de `iridium-extractor` vers un fichier `sortie.txt`. Ces trames successives doivent être analysées au moyen de `iridium-parser.py` de `iridium-toolkit` qui prend en argument l'option `-p` suivi du fichier enregistré par `iridium-extractor`. Ainsi

```
iridium-parser.py -p sortie.txt > sortie.parser
```

transforme les 4.2 GB d'enregistrements acquis en deux jours (52,85 heures exactement, limité par la taille du RAMfs allouée dans `/tmp` de la Raspberry Pi 5) en semaine, en 1850 MB de trames analysées de la forme

```
IRA: p-1710866743 000001123.6170 1626276841 100% -70.17|-128.72|19.24 158 DL sat:016
beam:40 xyz=(+1096,+0312,+1112) pos=(+44.30/+015.89) alt=013 RAI:48 ?00 bc_sb:26 P02:
PAGE(tmsi:93fc8914 msc_id:17) PAGE(tmsi:06fd50ce msc_id:25) {OK}
IDA: p-1710866743 000001202.4166 1625401811  93% -73.15|-127.78|24.39 179 DL
LCW(2,T:maint,C:switch[dtoa:0,dfoa:0],000)
...
```

Un graphique avec la nature des messages reçus en fonction du temps s'obtient par le script `stats.py` de `iridium-toolkit` prenant comme argument le fichier ainsi analysé. Pour un jour de semaine avec une antenne bien exposée au ciel (Fig. 4), le graphique est de la forme de celui proposé en Fig. 6, avec `stats.py` trivialement modifié pour rendre le graphique un peu moins hideux.

Parmi les nombreux messages capturés, nous trouvons ceux commençant par IRA pour "Iridium Ring Alert" et qui incluent les spécifications du faisceau micro-ondes qui nous illumine incluant la position du satellite dans le ciel. Ces informations sont analysées au moyen du script `https://github.com/muccc/iridium-toolkit/blob/master/mkkml` afin de produire une carte au format Keyhole (KML) par

```
grep ^IRA sortie.parser | perl mkkml tracks > sortie.kml
```

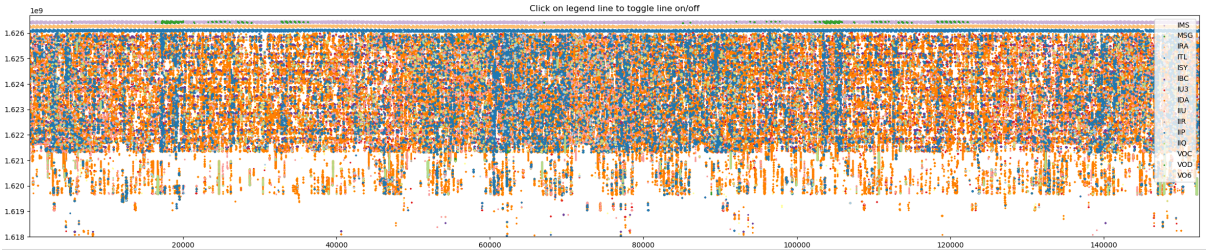


FIGURE 6 – Affichage des messages reçus, en abscisse le temps en secondes et en ordonnée la fréquence de la sous-porteuse sur laquelle le message a été reçu. Acquisitions obtenues par une Raspberry Pi 5 connectée à un récepteur Ettus Research B210 pendant une cinquantaine d'heures en semaine, , pour un fichier d'enregistrement de 4,2 GB.

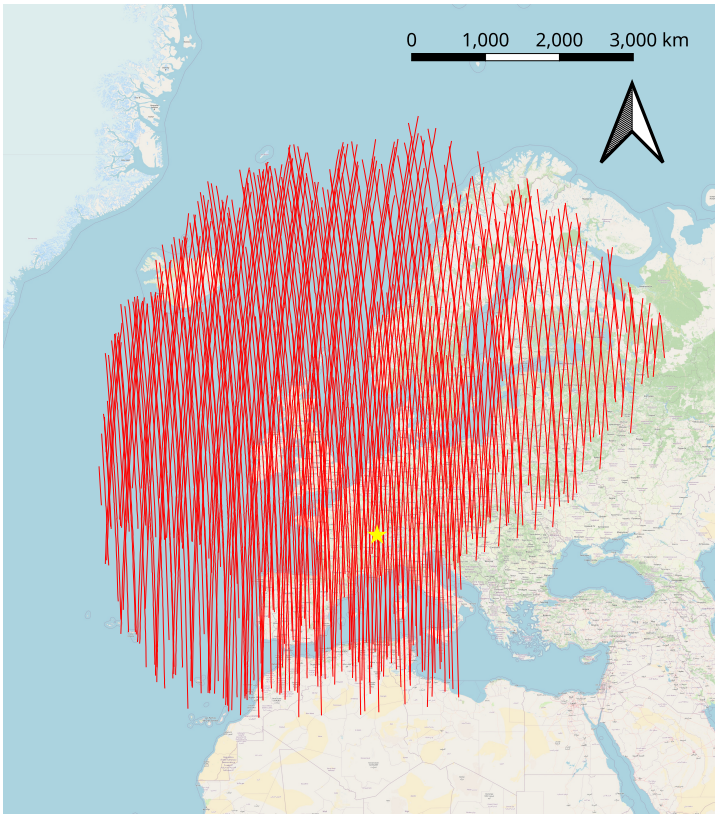


FIGURE 7 – Traces des satellites qui ont survolé le récepteur localisé à Besançon dans l'Est de la France (étoile), superposées dans QGis au fond de carte OpenStreetMaps. Les signaux ont été acquis depuis l'emplacement visible sur la Fig. 4, découvert vers l'Ouest et le Nord mais partiellement bloqué vers l'horizon sud-est, d'où l'asymétrie de la distribution. L'extension nord-sud des traces est de 4600 km et est-ouest de 4000 km, en accord avec la visibilité attendue d'un satellite survolant la Terre à 765 km d'altitude.

qui peut être introduit dans QGis pour être superposé sur un fond de carte, démontrant notre analyse antérieure d'un rayon de visibilité de 3000 km environ (Fig. 7).

Enfin nous pouvons nous atteler à décoder les messages utiles, et en particulier les messages transmis par les avions vers les aéroports au format ACARS ainsi que les messages vocaux et les messages textuels SMS. L'analyse des messages ACARS donne des résultats qui complémentent élégamment notre `gr-acars` [9] qui se contente de recevoir les messages transmis directement par les avions vers le sol. Ici nous obtenons des messages d'avions qui sont clairement au-delà de l'horizon auquel nous pourrions prétendre

recevoir un message directement (horizon qui vaut 375 km pour un avion volant à 11 km d'altitude selon l'analyse précédente'). Dans la Fig. 8, un avions décollant d'Amsterdam à 580 km de Besançon ne pourrait être observé en vue directe. Pour ce faire, `iridium-toolkit` fournit un script pour re-assembler les messages et décoder les trames de communication des avions selon un protocole que nous avons déjà largement décrit dans ces pages [10] : toujours sur le fichier qui a été analysé,

```
reassembler.py -i sortie.parser -m acars
```

fournit majoritairement des identifiants d'aéronefs que nous retrouverons sur FlightRadar24, à des distances bien au-delà de la vue directe du récepteur à l'avion, prouvant que c'est bien des trames relayées par satellite que nous obtenons, mais aussi quelques messages dont le sens est souvent obscur si le code utilisé n'est pas connu. Le fichier de 52,85 h d'acquisition contient 2163 trames ACARS décodées, dont 88 messages qui ne se limitent pas juste à un identifiant d'avion, de la forme

```
Dir:DL Mode:2 REG:HBIGO   NAK   Label:H1 (Message to/from terminal) bID:M [-
#MDMSG/RX19-MAR-24 16:43UTC/RX
/RXSATCOM DIRECT ALERT/RX SATCOM DIRECT ALERT FOR HBIGO/RXROUTE ALERT
NOTIFICATION:/RXYOUR FLIGHT PLAN L6082 EDDB-LSZH HAS BEEN EVALUATED. CURRENTLY,
THERE ARE NO ALERTS./RX82ED]
```

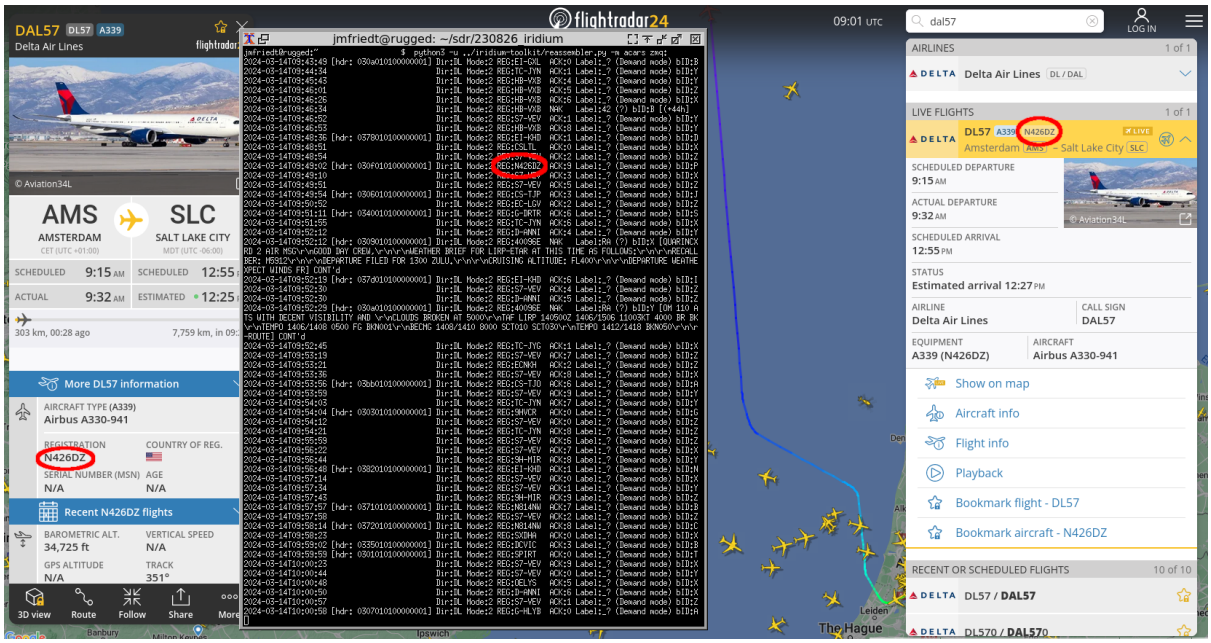


FIGURE 8 – Analyse des trames ACARS reçues par Iridium en comparant les identifiants des aéronefs dans FlightRadar24 pour localiser les avions au moment de la réception du message, bien au-delà de l'horizon pour espérer une vue directe depuis notre site de réception, prouvant que le signal vient bien d'un satellite.

Enfin Iridium étant conçu pour transmettre initialement de la téléphonie, nous pourrions vouloir écouter les conversations. Toujours dans `iridium-toolkit`, les outils pour ce faire sont

```
stats-voc.py sortie.parser
```

qui a nécessité au préalable de mettre en place les outils pour décoder le codec audio propriétaire :

```
sudo cp iridium-toolkit/play-iridium-ambe /usr/local/bin/
sudo cp iridium-toolkit/bits_to_dfs.py /usr/local/bin/
```

La majorité des messages vocaux décodés vient de boîtes de réception automatiques, et seulement quelques conversations brèves sont audibles, souvent limitées à quelques secondes, peut être compte tenu du tarif de la liaison Iridium. Depuis Besançon, la majorité des conversations sont en allemand, et seuls quelques échanges en français ont été audibles (Fig. 9.

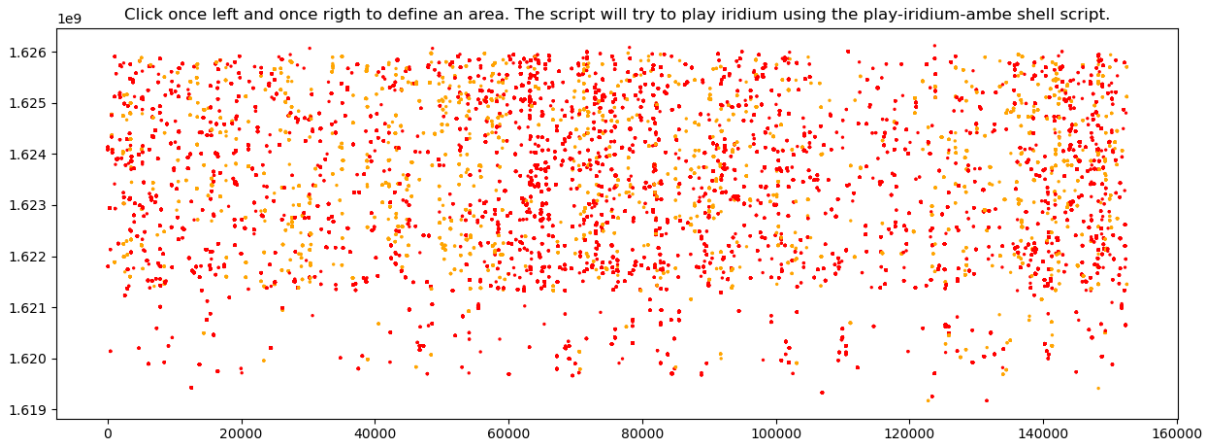


FIGURE 9 – Interface graphique pour sélectionner chaque conversation et en écouter le contenu : l'utilisateur clique (bouton gauche puis bouton droit) sur un rectangle qui contient un certain nombre de points rouge, et chacun contient un message qui sera décodé et joué sur le carte son. La graduation des axes n'étant pas le point fort des auteurs de `iridium-toolkit`, l'abscisse reste le temps en secondes et l'ordonnée la fréquence du canal portant une information en hertz.

5 Messages textuels

Bien entendu si nous parlons de GSM nous pensons SMS et donc messages transmis au travers des satellites. En plus du protocol ACARS, `iridium-toolkit` est aussi capable de décoder les paquets de la couche 3 GSM [11] dont les SMS communiqués depuis l'espace vers un récepteur au sol "pas trop loin" (moins de quelques centaines de kilomètres) de la radio logicielle à l'écoute d'Iridium.

Le décodage des messages par

```
iridium-toolkit/reassembler.py -i sortie.parser -m msg
```

est décevante car ne comprend que des messages d'apparence chiffrée. Ce n'est pas le travail de `gr-iridium` de décoder les messages, mais comme souvent avec GNU Radio un outil externe fera mieux l'affaire. Dans ce cas, [12] nous enseigne d'installer `tshark` – disponible en paquet binaire pour Debian/GNU Linux (`apt install tshark`) – qui prend en entrée la séquence des paquets capturés et décodés par

```
iridium-toolkit/reassembler.py -i sortie.parser -m lap # produit sortie.pcap
```

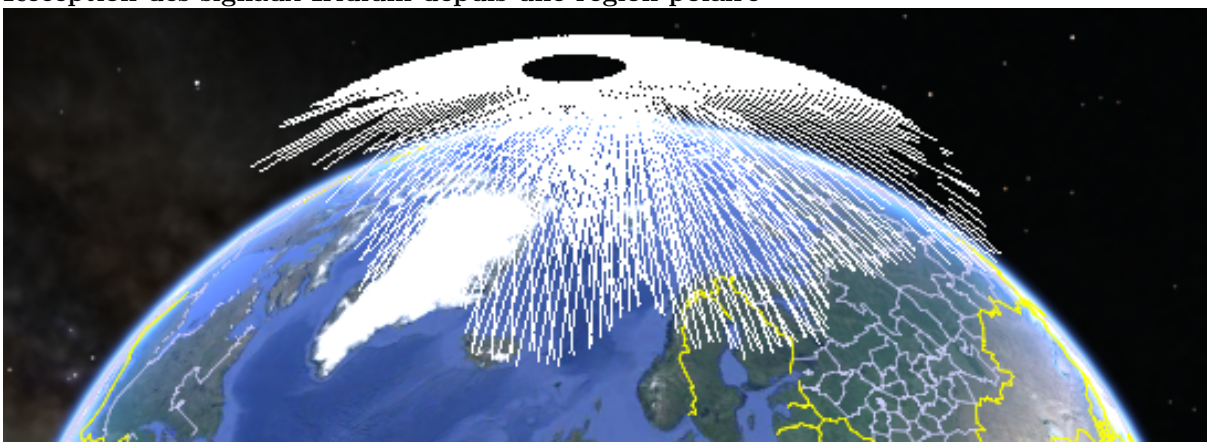
que Sec et Schneider décrivent comme "*GSM-compatible L3 messages as GSMtap compatible .pcap*". Ce fichier est analysé par `tshark` en ne proposant que les champs lisibles par un humain

```
tshark -r sortie.pcap -Y gsm_sms -T fields \-e gsm_sms.tp-oa -e gsm_sms.sms_text
```

dont nous nous retenons de reproduire la sortie dans ces pages puisque contenant des numéros de téléphone et les messages (fort peu intéressants au demeurant avec un contenu informatif aussi utile que "essai tel sat") associés en clair ... mais qui contiennent tout de même la mention "*This message contains information which may be confidential*". Il faudrait peut-être que NetJets pense à changer de mode de communication dans ce cas. L'ouverture du fichier `pcap` dans `wireshark` donne comme d'habitude accès à tous les champs clairement décodés et annotés du protocole GSM.

FIGURE 10 – Gauche : montage expérimental, avec l'antenne GPS modifiée pour la réception d'Iridium en vue du ciel (et des glaciers à front marin de Kongsfjorden), le téléphone incitant les satellites à communiquer vers notre site, et milieu-droite les messages textuels reçus par le téléphone.

Réception des signaux Iridium depuis une région polaire



La génération du fichier KML représentant les points où se trouvent les satellites au moment de la réception du signal et l'affichage dans Google Earth permet de visualiser la distance ridiculement faible qui sépare les satellites Iridium survolant la surface de la Terre à près de 800 km par rapport au rayon terrestre de 6400 km. Par ailleurs, la réception depuis un site proche des pôles de signaux transmis par des satellites en orbite polaire illustre le problème d'empilement des faisceaux radiofréquences quand tous les satellites passent dans la même zone, un problème explicitement mentionné dans [1] et qu'il fallut résoudre pour que la constellation Iridium fonctionne correctement.

Afin de valider notre compréhension de l'échange de messages textuels (SMS) par Iridium, nous avons profité d'une mission de terrain en région arctique norvégienne (Fig. 10) pour tester l'émission de SMS depuis un téléphone mobile et sa réception par Iridium. En effet, l'Institut Paul-Émile Victor (IPEV) nous fournit sur place un téléphone Iridium pour la sécurité lors des déplacements, tandis que la station scientifique de Ny-Ålesund est équipée depuis fin 2023 d'un réseau 4G sub-GHz (pour ne pas perturber excessivement la réception du radiotélescope présent sur site) en données uniquement [13]. L'expérience se poursuit donc comme suit :

— nous avons emporté au fond de notre sac une Pluto+ et le T de polarisation avec l'antenne patch modifiée pour recevoir Iridium. La chaîne de traitement illustrée en Fig. 11 permet d'acquérir le flux de données depuis la Pluto+ dans GNU Radio et émettre les trames par Zero-MQ tel que nous l'avons vu auparavant. Cette acquisition se poursuit en continu, produisant environ 1 GB/jour de données brutes,

— le téléphone Iridium, dont le numéro d'appel est connu, est allumé et connecté au réseau satellitaire pour induire des communications espace-sol et informer la constellation de satellites de l'emplacement du récepteur. Rappelons que les satellites Iridium sont munis d'antennes directives qui ne rayonnent *que* vers le récepteur – dans un disque de rayon 200 km autour du récepteur d'après `https://www.satellitephonereview.com/iridium-coverage/` – et pas vers une surface importante comme le fait un satellite géostationnaire de communication muni seulement d'une parabole de taille modeste. Le récepteur de radio logicielle doit donc être à une distance inférieure de 400 km du téléphone Iridium,
— un message court (SMS) est émis depuis le téléphone mobile (ou depuis le satellite Iridium) vers la station de base 4G qui l'injecte dans le réseau de communication jusqu'au réseau Iridium qui le communique vers ses satellites qui, se relayant l'un vers l'autre, arrive jusqu'au satellite en vue du téléphone et émet le message,
— la Pluto+ écoute en continu les signaux descendants et capture les trames brutes par `iridium-extractor -D 4 --multi-frame ./examples/zeromq-sub.conf > fichier.txt` comme nous l'avons vu auparavant,
— ces trames sont traitées par `iridium-toolkit/iridium-parser.py --harder --uw-ec fichier.txt > fichier.parser` qui cette fois ne se contente pas de ne conserver que les trames complètes (option `-p` de l'utiisation de cette commande auparavant) mais essaie de profiter des corrections d'erreur,
— le fichier résultat est utilisé pour trouver les trames GSM par `iridium-toolkit/reassembler.py -i fichier.parser -m lap`
— finalement, les textes de SMS sont recherchés par `tshark -r fichier.pcap -Y gsm_sms -T fields \-e gsm_sms.tp-oa -e gsm_sms.sms_text`

dont le résultat sur une après-midi d'acquisitions est

```
       1:L:78.94538:12.02204:1:240430114817:280:6.3
33633XXXXXX     Hello world jmfriedt
       1:L:78.89683:12.30865:1:240430120510:323:0.2
       1:L:79.05680:11.53959:1:240430124817:356:9.6
33633XXXXXX     Hello world jmfriedt
00881662XXXXXX  Essai a 16h28
...
       1:L:78.92872:11.93505:1:240430155249:268:0.0
...
```

L'origine du deuxième et cinquième message sont suffisamment clairs : il s'agit des SMS que nous avons émis depuis le téléphone portable et reçu par Iridium. Nous avons donc convenablement compris le fonctionnement du système de communication. Les premier, troisième et quatrième message font partie d'une séquence qui se reproduit régulièrement et dont les premières coordonnées GPS se trouvent toujours dans le fjord de la baie du Roi à proximité de Ny-Ålesund : nous les attribuons aux nombreux bateaux de touristes fortunés qui viennent admirer les glaciers à front marin de ce paysage exceptionnel. Il est probable que ces navires échangent leur position par Iridium dans les zones hors couverture de réseaux cellulaires et en compléments des communications AIS autour de 162 MHz qui ne portent que sur quelques centaines de kilomètres (Fig. 12). Le dernier message ... n'est pas de nous mais est le début d'une série de messages émis depuis un terminal Iridium. Tous les numéros ont été anonymisés en remplaçant leurs chiffres de poids faibles par des "X".

Deux erreurs que nous avons faites au cours de ces expériences : tout d'abord, la Pluto+ ne peut que recevoir la moitié de la bande Iridium (5 MHz). Nous avions mal compris `https://github.com/muccc/gr-iridium` en pensant que les SMS sont des messages unidirectionnels transmis dans la moitié supérieure de la bande ($1624,5 \pm 2,5$ MHz) mais il n'en est rien, le succès fut atteint lors de l'écoute de la bande inférieure (($1620,5 \pm 2,5$ MHz). Par ailleurs, le téléphone répond au satellite avec un signal relativement puissant qui aveugle la Pluto+ le temps de sa liaison montante, il vaut donc mieux éloigner le téléphone Iridium de la Pluto+ de quelques kilomètres pour éviter de perturber la réception.

Parmi les messages reçus au hasard des écoutes, "7967894XXXX Домой езжай они седня приедут" est difficile à interpréter [1] mais a certainement été reçu par un téléphone dans un rayon d'une centaine de kilomètres de Ny-Ålesund, la Russie (préfixe 79) ayant un certain nombre de villes installées au Svalbard, en accord avec le traité de Paris de 1920 [14].

1. Google Translate traduit par "Rentre chez toi, ils arriveront aujourd'hui "

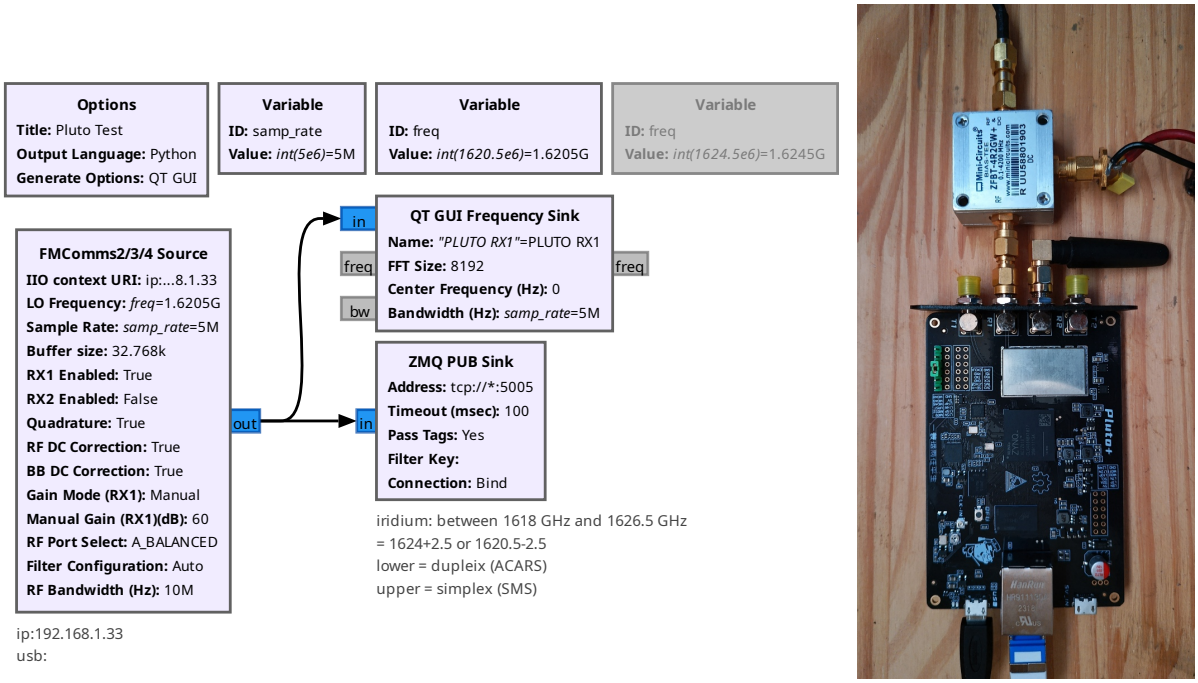


FIGURE 11 – Gauche : chaîne d'acquisition de la Pluto+ : noter en commentaire la fréquence centrale de 1624,5 MHz qui permet à la Pluto+ de recevoir la moitié supérieure de la bande Iridium mais qui n'est *pas* celle qui contient les SMS, tandis que le bloc décommenté centre l'oscillateur local sur 1620,5 MHz et la bande passante sur le maximum accessible sans perte de données de 5 MHz. Droite : Pluto+ et son T de polarisation pour alimenter le préamplificateur au plus proche de l'antenne dans la base Jean Corbel de l'IPEV, 78, 9°N. Seule l'entrée gauche est utilisée pour Iridium, l'antenne 868 MHz sur l'entrée droite sert de détrompeur et pour observer les trames du réseau de capteurs LoRa.

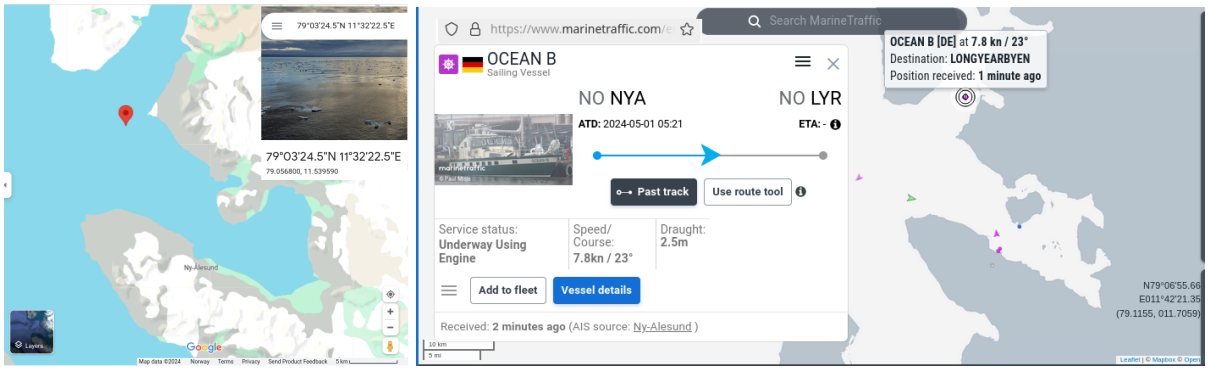


FIGURE 12 – Gauche : position d'une des coordonnées reçues par Iridium dans le fjord de la baie du Roi – Kongsfjorden – et droite : capture d'écran de `marinetraffic.com` autour de la même date, qui a acquis les positions annoncées des navires par AIS et les retranscrit sur une carte.

6 Utilisation sur système embarqué

Étant donné que Iridium est capable de focaliser, par synthèse électronique d'antenne, le faisceau émis par un satellite vers le destinataire sans illuminer une surface excessivement importante, il semble nécessaire de déployer un système d'écoute "proche" de la cible surveillée afin de recevoir les communications qui lui sont destinées. Compte tenu de la puissance de calcul nécessaire pour traiter le flux de données issues des radio logicielles, il semble évident qu'un microcontrôleur ne répondra pas au besoin mais qu'en est-il des Raspberry Pi ? Nous avons déjà exploré les performances des ordinateurs monocartes à base

de processeur ARM [15] et avions conclu que le jeu d'instructions SIMD NEON est considérablement moins performant que ceux fournis pas Intel/AMD, mais essayons tout de même. Buildroot supporte la Raspberry Pi 4, mais pas officiellement [2] la Raspberry Pi 5, qui se contentera donc d'une distribution binaire Raspberry Pi OS issue de Ubuntu (Fig. 13).

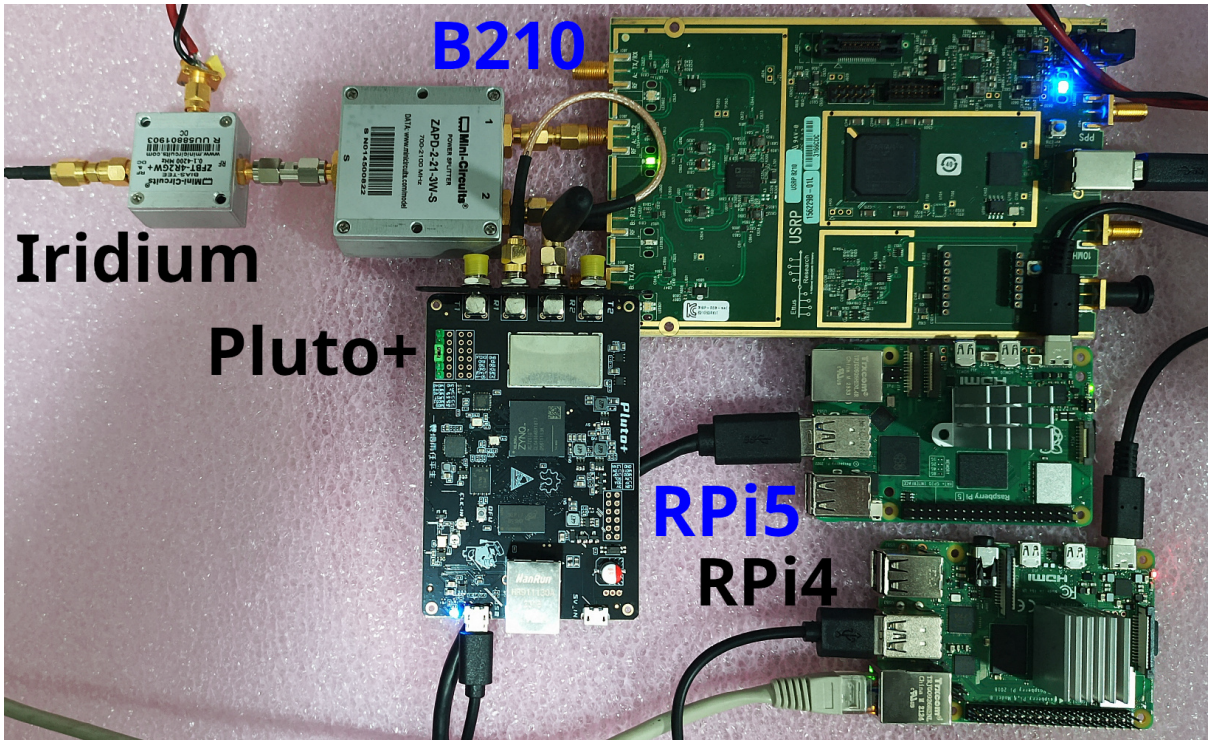


FIGURE 13 – Photo de famille des plateformes embarquées envisagées : le Raspberry Pi 4 est connectée par USB à la Pluto+ puisque les deux ne peuvent produire ou traiter plus de 5 Méchantillons/seconde, et la Raspberry Pi 5 est connectée par USB à l'Ettus Research B210. La stabilité du lien GNU Radio - ZeroMQ - `gr-iridium` sur Raspberry Pi 4 s'est avérée décevante, tandis que la Raspberry Pi 5 a fonctionné sans broncher une journée jusqu'à remplir son RAMfs de données.

La cross-compilation de GNU Radio dans Buildroot souffrait encore récemment d'un défaut bien identifié, mais qui semble résolu dans 2024.02 et le passage à une version supérieure à 2.11 de `pybind`, que le suffixe de l'hôte et non de la cible est ajouté en fin de nom de chaque bibliothèque dynamique, qui ne sera donc pas trouvée au moment de l'exécution. Nous renommons le cas échéant à la main pour le moment toutes les bibliothèques (`.so`) contenant `-x86_64` en `aarch64` :

```
mv libpyuhd.cpython-311-x86_64-linux-gnu.so libpyuhd.cpython-311-aarch64-linux-gnu.so
...
```

et il en sera de même pour la cross-compilation de `gr-iridium` puisque

```
mv /usr/lib/python3.11/site-packages/iridium/iridium_python.cpython-311-x86_64-linux-gnu.so \
   /usr/lib/python3.11/site-packages/iridium/iridium_python.cpython-311-aarch64-linux-gnu.so
```

et malgré tous les efforts de Guillaume Bressaix pour inclure SciPy dans Buildroot, nous n'avons pas réussi à le compiler. Qu'à cela ne tienne, la dépendance à `scipy` dans `python/iridium_extractor_flowgraph.py` est inutile et peut simplement être retirée [3] sans impacter le résultat du calcul. On pensera à passer l'ordinateur monocarte – Raspberry Pi 4 ou 5 – en performance

2. nous avons tenté la compilation par le dépôt Buildroot pour Raspberry Pi 5 disponible à `https://github.com/home-assistant/operating-system/` sans succès mais sans avoir trop persévéré il faut bien avouer
3. les auteurs de `iridium-toolkit` ont appliqué cette correction à notre demande

```
echo "performance" > /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
```

et à calibrer VOLK en exécutant une fois `volk_profile` afin de bénéficier des instructions NEON de l'ARM. L'utilisation des instructions SIMD est fondamentale pour atteindre les performances visées puisque

```
# volk_profile
RUN_VOLK_TESTS: volk_64u_popcntpuppet_64u(131071,1987)
generic completed in 3192.56 ms
neon completed in 1976.58 ms
Best aligned arch: neon
Best unaligned arch: neon
RUN_VOLK_TESTS: volk_16u_byteswappuppet_16u(131071,1987)
generic completed in 850.555 ms
neon completed in 287.825 ms
neon_table completed in 319.742 ms
Best aligned arch: neon
Best unaligned arch: neon
```

montre bien des gains d'un facteur de plus de trois sur la vitesse d'exécution de traitements simples mais parallélisables. Malgré tous ces efforts, la Raspberry Pi 4 ne peut pas traiter 10 Méchantillons/seconde : elle perd 70% des paquets acquis par la radio logicielle sans NEON, pour passer à 30% avec NEON. On peut gagner un peu en *overclockant* le processeur de 1500 à 2300 MHz mais il y a toujours une quinzaine de pourcents de paquets perdus. Nous devons donc nous résigner à abaisser la fréquence d'échantillonnage des signaux acquis par Raspberry Pi 4 à 5 MHz en ajustant la fréquence d'échantillonnage du fichier de configuration `https://github.com/muccc/gr-iridium/blob/master/examples/usrp-b2x0-uhd.conf`. Ainsi, la Raspberry Pi 4 arrive à acquérir un nombre décent de paquets et en traiter la majorité :

```
1710756542 | i: 143/s | i_avg:  86/s | q_max:   16 | i_ok:  63% | o:  338/s | ok:  71% | ok: 102/s | ok_avg:  58% | ok:
1710756543 | i: 152/s | i_avg:  96/s | q_max:   15 | i_ok:  83% | o:  376/s | ok:  92% | ok: 141/s | ok_avg:  66% | ok:
1710756544 | i: 163/s | i_avg: 104/s | q_max:   19 | i_ok:  68% | o:  402/s | ok:  79% | ok: 130/s | ok_avg:  69% | ok:
1710756545 | i: 174/s | i_avg: 112/s | q_max:   20 | i_ok:  75% | o:  409/s | ok:  82% | ok: 143/s | ok_avg:  71% | ok:
1710756546 | i: 172/s | i_avg: 118/s | q_max:   17 | i_ok:  71% | o:  389/s | ok:  78% | ok: 134/s | ok_avg:  72% | ok:
...
```

dont le résultat est stocké en RAMfs pour ne pas détériorer la carte SD et ne pas être limité par son débit de communication. La référence [16] est intéressante en ce qu'elle explique chaque champ et comment analyser leur pertinence. Ce tableau se lit comme suit :

1. la première colonne est le temps unix donc en secondes depuis le 1er janvier 1970,
2. la deuxième colonne est le nombre de messages reçus à un instant donné, donc d'autant plus élevé que l'antenne est de bonne qualité, bien placée et que le gain du récepteur est convenablement ajusté (et qu'il y a du traffic sur le satellite Iridium passant au-dessus du site),
3. la troisième colonne est une moyenne glissante de la deuxième,
4. la quatrième colonne `q_max` indique le nombre de message empilés dans la queue en attente d'être décodés : plus ce nombre est grand, plus le(s) processeur(s) est/sont surchargés et n'arrivent pas à suivre la cadence des messages,
5. la cinquième colonne `i_ok` le nombre de messages utiles et décodés,
6. la dixième colonne est le nombre de messages effectivement décodés depuis le début de `iridium-extractor`, avec de nouveau la 11ème colonne une moyenne glissante de celle-ci,
7. finalement la dernière colonne est le nombre de messages qu'il a fallu laisser tomber faute de puissance de calcul.

Noter que la Raspberry Pi 4 arrive ici à suivre la cadence *car nous avons limité la bande passante* à 5 MHz, mais n'arrive pas à en faire autant avec 10 MHz pour lesquels au-moins 15% des paquets radiofréquences issus de la B210 sont perdus.

7 Communication inter-satellites

Les satellites géostationnaires américains TDRS et européens EDRS sont au cœur de la capacité à relayer des informations des satellites en orbite basse, avec leur empreinte au sol réduite, vers les stations de réceptions au sol après des sauts multiples [17]. Alors que faire communiquer des satellites géostationnaires entre eux peut sembler compréhensible compte tenu de leur position à peu près fixe dans le ciel, les Iridium circulent à plus de 7,5 km/s et pourtant arrivent à s'échanger des informations via des faisceaux micro-ondes. La constellation Starlink évolue rapidement et alors que les premiers satellites n'avaient pas la capacité à communiquer entre eux, il semble qu'actuellement les satellites puissent échanger des informations au travers de liaisons optiques, tout comme EDRS [18]. Nous avons mentionné auparavant que cette fonctionnalité est la source de l'anonymisation de l'origine des appels, mais n'empêche pas le réseau de s'effondrer lors d'un nombre excessif d'appels sur une cellule [1] :

"*It seemed that satellites in one of the six orbital planes had been stressed beyond capacity.*

But that was impossible. The Iridium system could handle 98,586 calls at a time – the math had been worked out by Ken Peterson long ago – and the company barely had 140,000 phones in service. Could there be a glitch in the pattern of cell reuse ? Could the phased-array antennas be duplicating the same call or failing to pass it off to the next satellite ? The technician dug deeper into the numbers scrolling rapidly across his screen, zeroed in on the heavy traffic, and then realized all the calls were originating in the same fifty-thousand-square-mile zone. It was possible to overload one small part of the system if, by some odd set of circumstances, massive numbers of people were all sending signals to the same satellite.

And that's what was happening. It was a holiday in the United States, and there were 146,000 American troops in Iraq, and there was only one phone they could all depend on. Technically you weren't supposed to use an Iridium phone for a "morale call," but when you absolutely, positively had to get your call through at a certain time to a certain person, every soldier, sailor, Marine, and airman knew there was only one solution. The only situation that could max out the Iridium system was Mother's Day in a combat zone. "

8 Conclusion

Alors que les satellites en orbite géostationnaire proposent une foultitude de signaux à décoder et analyser, la prolifération des constellations de satellites en orbite basse les rend plus favorables aux communications cellulaires depuis le sol et donc fuites d'informations si leur signal descendant peut être décodé (on ne peut plus parler d'"intercepter" pour une liaison radiofréquence ouverte à tous pour réception). Nous avons illustré ce principe sur Iridium en démontrant l'exploitation de `gr-iridium` s'appuyant sur diverses plateformes matérielles d'acquisition et de traitement, éventuellement embarquées pour déploiement proche du site d'intérêt. Après des débuts commerciaux difficiles, Iridium NEXT a été assemblé par Thales Alenia Space et lancé pour poursuivre les services, avec l'ajout notamment de la promesse de la dissémination du temps, vitesse et position (PNT) depuis un satellite en orbite basse moins sensible aux interférences que GPS/Galileo/GLONASS/Beidou [19]. Cependant, Iridium propage aussi un signal de navigation et de temps nommé STL [20] qu'il serait certainement intéressant d'ajouter à la panoplie des messages déjà analysés par `gr-iridium` pour ne pas uniquement s'appuyer sur le décalage Doppler observé depuis le sol pour se localiser.

Iridium reste dans l'imaginaire populaire un mythe associé à la sécurité et aux environnements hostiles tel qu'en atteste `https://www.iridiummuseum.com/exhibits/iridium-in-the-spotlight/` et la réception de ces signaux ne peut que conforter dans cette appréciation.

Remerciements

L'Institut polaire Paul-Émile Victor (IPEV) a financé le déplacement au Spitsberg au cours duquel un certain nombre des mesures, acquises en dehors du cadre professionel de la mission de terrain, présentées dans cet article ont été effectuées, et a fourni malgré lui les outils pour l'expérimentation.

Références

[1] J. Bloom, *Eccentric Orbits : The Iridium Story*, Open Road+ Grove/Atlantic (2016)

[2] J.-M Friedt, *Le temps et son transfert par satellite géostationnaire : réception avec une parabole de télévision et d'une radio logicielle*, GNU/Linux Magazine France Hors Série **121** (2022)

[3] Telesat, *Telstar 11N Technical Manual, 2008 Edition*

[4] Kratos Defense, `https://www.kratosdefense.com/products/space/signals/rf-management/satid`

[5] *Eutelsat accuses Iran of jamming its satellites Reuters (2022)* à https ://www.reuters.com/world/eutelsat-accuses-iran-jamming-its-satellites-2022-10-06/ ou plus ancien, *France Seeks ITU Help To Halt Satellite Signal Jamming by Iran* (2010) à https ://spacenews.com/france-seeks-itu-help-halt-satellite-signal-jamming-iran/

[6] *Iridium backgrounder* à `https://www.winlab.rutgers.edu/~narayan/Course/Wireless_Revolution/LL24-%20IRIDIUM.pdf`

[7] Sec & Schneider, *Hacking Iridium*, Chaos Communication Camp (2015) à `https://www.youtube.com/watch?v=ahZOGhV8qnc`

[8] Sec & Schneider, *Iridium reverse engineering update*, OsmoDevCall à `https://people.osmocom.org/laforge/OsmoDevCall/20220325-Iridium-Update-Public.pdf` (2022)

[9] J.-M Friedt `https://sourceforge.net/projects/gr-acars/` (accédé Mars 2024)

[10] J.-M Friedt, *Bitstream clock synchronization in an ACARS receiver*, SDR-Academy online (2020), à `https://www.youtube.com/watch?v=54URhrJkk28`

[11] D.F. Kune, J. Koelndorfer, N. Hopper & Y. Kim, *Location leaks on the GSM air interface*, Proc. ISOC NDSS (Fév. 2012) à `https://www-users.cse.umn.edu/~hoppernj/celluloc.pdf`

[12] Slug309, *Extracting GSM/Cellular SMS Messages With Iridium-Toolkit And DragonOS* à `https://www.youtube.com/watch?v=GsgS3d0V5zc` (2022)

[13] T. Nilsen, *World's northernmost research settlement gets mobile phone service*, 27 Nov. 2023 à `https://thebarentsobserver.com/en/arctic/2023/11/worlds-northernmost-research-settlement-gets-mobile-phone-service`

[14] *Treaty between Norway, The United States of America, Denmark, France, Italy, Japan, the Netherlands, Great Britain and Ireland and the British overseas Dominions and Sweden concerning Spitsbergen signed in Paris 9th February 1920* à `http://library.arcticportal.org/1909/1/The_Svalbard_Treaty_9ssFy.pdf` qui mentionne "permit Russia to adhere to the present Treaty, Russian nationals and companies shall enjoy the same rights as nationals of the High Contracting Parties"

[15] G. Goavec-Merou, J.-M Friedt, *"On ne compile jamais sur la cible embarquée" : Buildroot propose GNU Radio sur Raspberry Pi (et autres)*, Hackable **37** (2021)

[16] *Iridium ACARS Decoding* à `https://thebaldgeek.github.io/Iridium.html` a disparu alors que nous rédigions cette prose mais a heureusement été archivé à `https://web.archive.org/web/20240206015959/https://thebaldgeek.github.io/Iridium.html`

[17] A. Marklund, *Inter-satellite links : Why it's important to expand usage in available spectrum*, ITU News Magazine -– Satellite connectivity, **4** p.24 (2023) à `https://www.itu.int/hub/publication/s-gen-news-2023-4/`

[18] A.U. Chaudhryand & H. Yanikomeroglu, *Laser Inter-Satellite Links in a Starlink Constellation : A classification and analysis*, IEEE Vehicular Technology Magazine, **16**(2), 48-56 (2021)

[19] H. Benzerrouk, *Iridium Next LEO Satellites as an Alternative PNT in GNSS Denied Environments -– Part 1*, June 17, 2019 à `https://insidegnss.com/iridium-next-leo-satellites-as-an-alternative-pnt-in-gnss-denied-environments-part-1/`

[20] Satellite Time and Location (STL) Technology (2019), à `https://www.everythingrf.com/community/satellite-time-and-location-stl-technology`