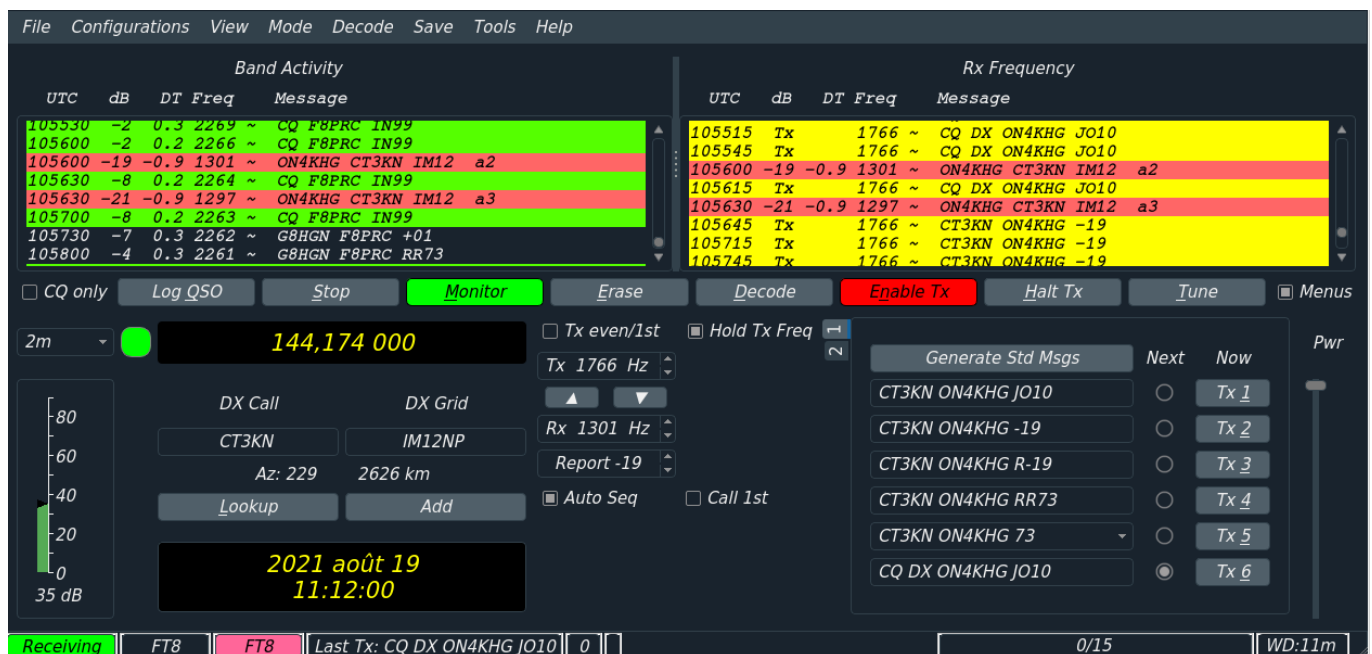


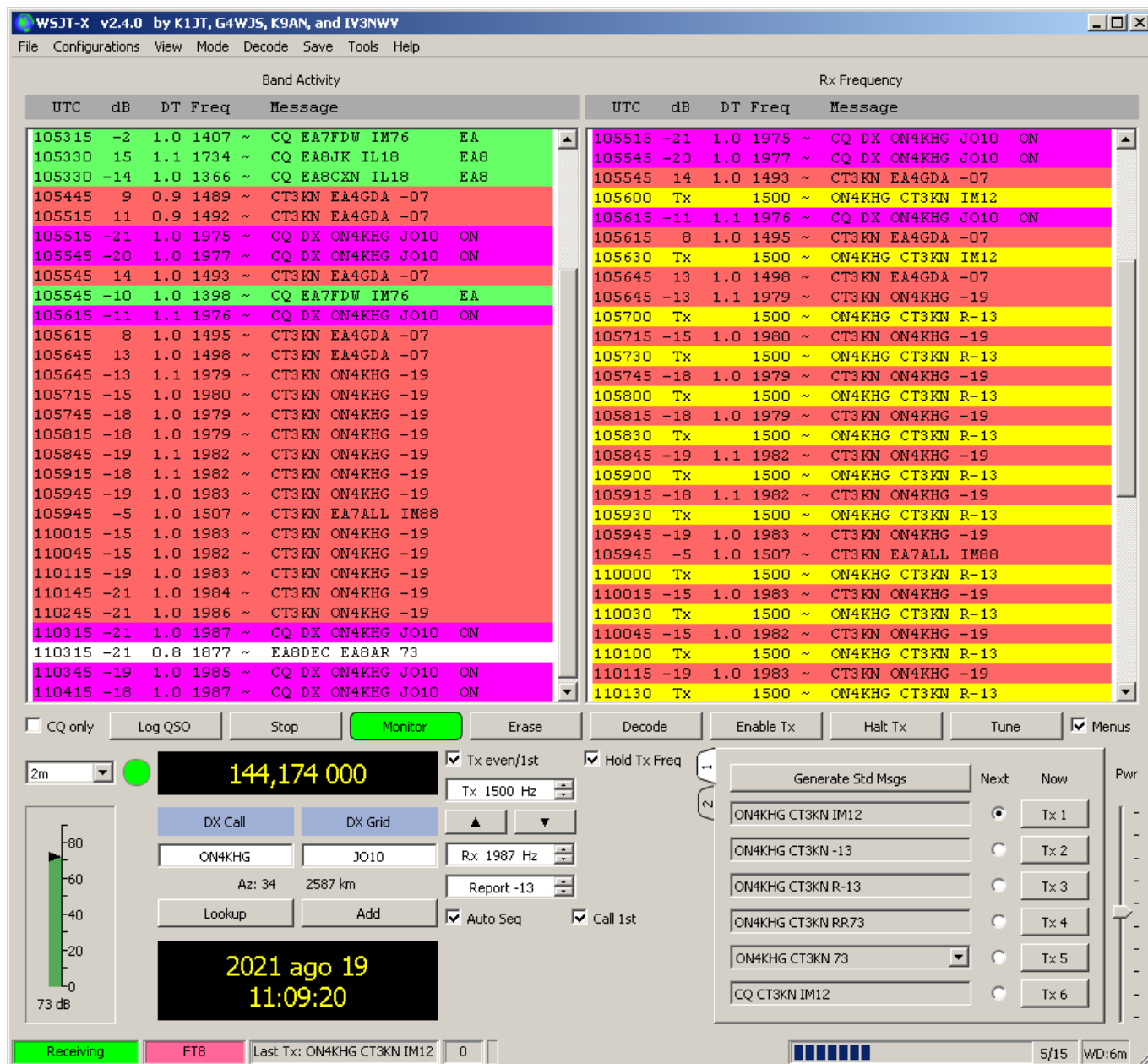
CT3 <> ON Tropo path on 144 MHz

On August 19th, 2021, I had the surprise to see “CT3KN” (IM12MT) on my screen in FT8 on 144 MHz at 10:56 UTC. Unfortunately, we couldn’t make a 2 ways QSO over that 2614 km path, 2-3 dB were missing to achieve it ! In 2011 and 2019, I already worked the Azores Islands (CU8) on 144 MHz in Tropo. Nevertheless, I never thought the path to CT3 (Madeira Island) would ever been possible. Indeed, if the sea propagation seems sometimes without “limits” (see the QSO’s on 144 MHz between D4 and EA8 and the Caribbean Islands), when there is land in between, that is another story. If on the path to the Azores the amount of land is very limited, on the path to Madeira, there is about 750 km of land to cross, with as many opportunities to have the tropo sea duct being interrupted.

Nevertheless, knowing that G stations as close to me as J001/02 made their way to the Ricardo (CT3KN)’s log on August 18th evening, I called “CQ” from time to time with the antennas heading to the Azores on the 19th. And it paid ! Below a screenshot taken at my station :



And here is the screen at Ricardo’ side at the same time :



Ricardo is using 80W and 2x9 elements antennas. I'm using 1,2 kW and 2x9 elements too. It makes a difference of more than 10 dB, which is somehow reflected in the reports seen. I saw Ricardo -19 dB at best and he saw me -11 at best. At -11 dB, my signal was barely audible in a reduced bandwidth. Hence, even in CW, a QSO would have been hard to achieve (at same RF power level both sides).

In the morning of August 19th, 2 EA1 stations lying on the path (both in IN73DM) have been worked too. EB1B has been worked at 07:27 UTC and EB1FNS at 09:42 UTC.

After analysis of the PSK Reporter data and my log, one can see :

- 06:22 UTC : CT2HXM (IN60CR) sees my signal -12 dB. Thierry sees also G0MBL in J001 at 07:23 UTC, nobody else around or in between. This can be MS or

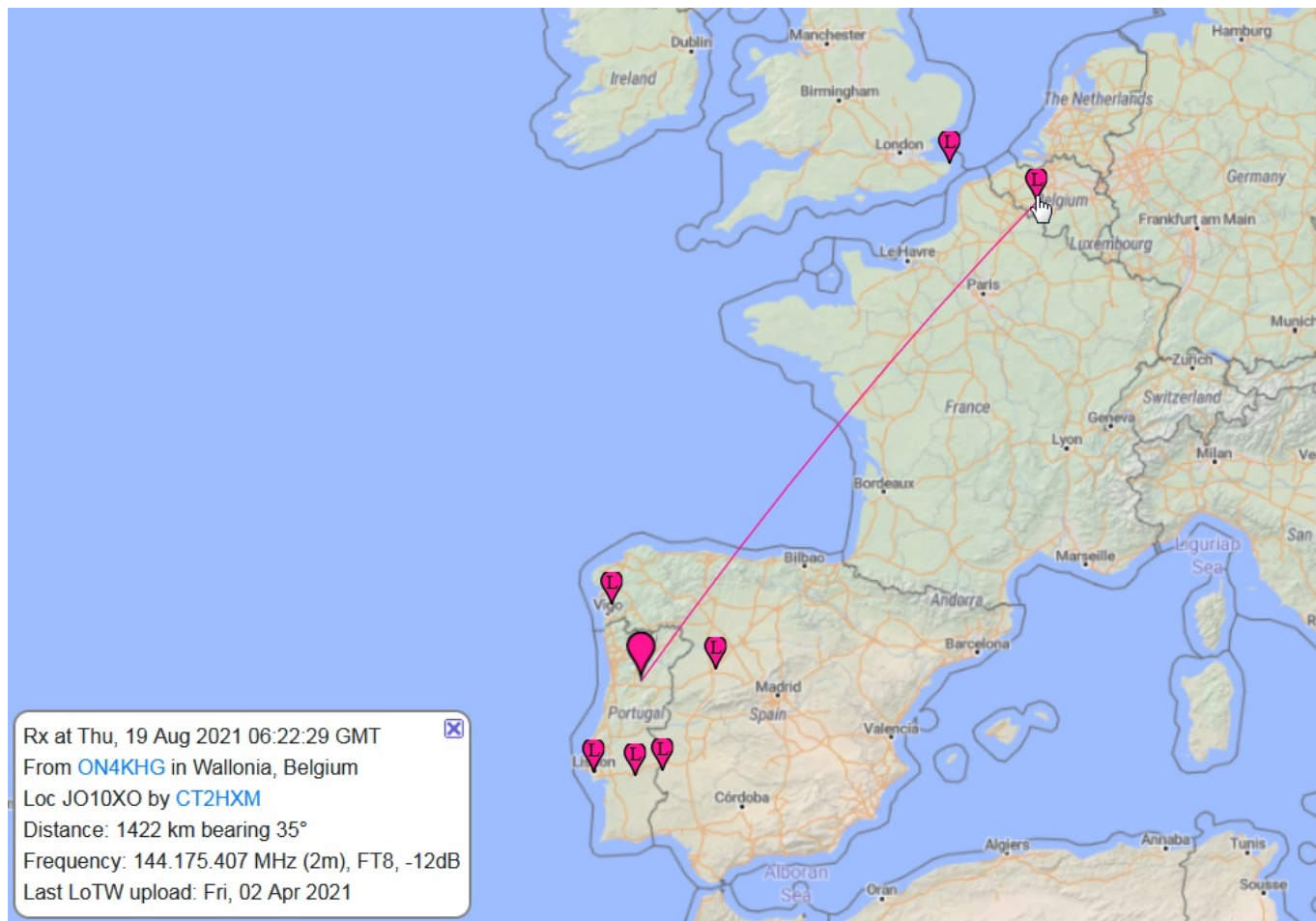
Tropo but I assume it was more MS than Tropo. See the map below.

- 07:27 UTC : I make QSO with EB1B in IN73DM (-08 dB / -08 dB)
- 09:42 UTC : I make QSO with EB1FNS in IN73DM (-08 dB / -01 dB)
- 10:56 UTC : uncomplete QSO with CT3KN in IN12MT (2614 km)
- 11:40 UTC EB1FNS sees my signal +15 dB. So, the duct seems to be stronger now, provided the antenna at the EB1FNS' side was heading the same QTF (to me ?) at 09:42 too.

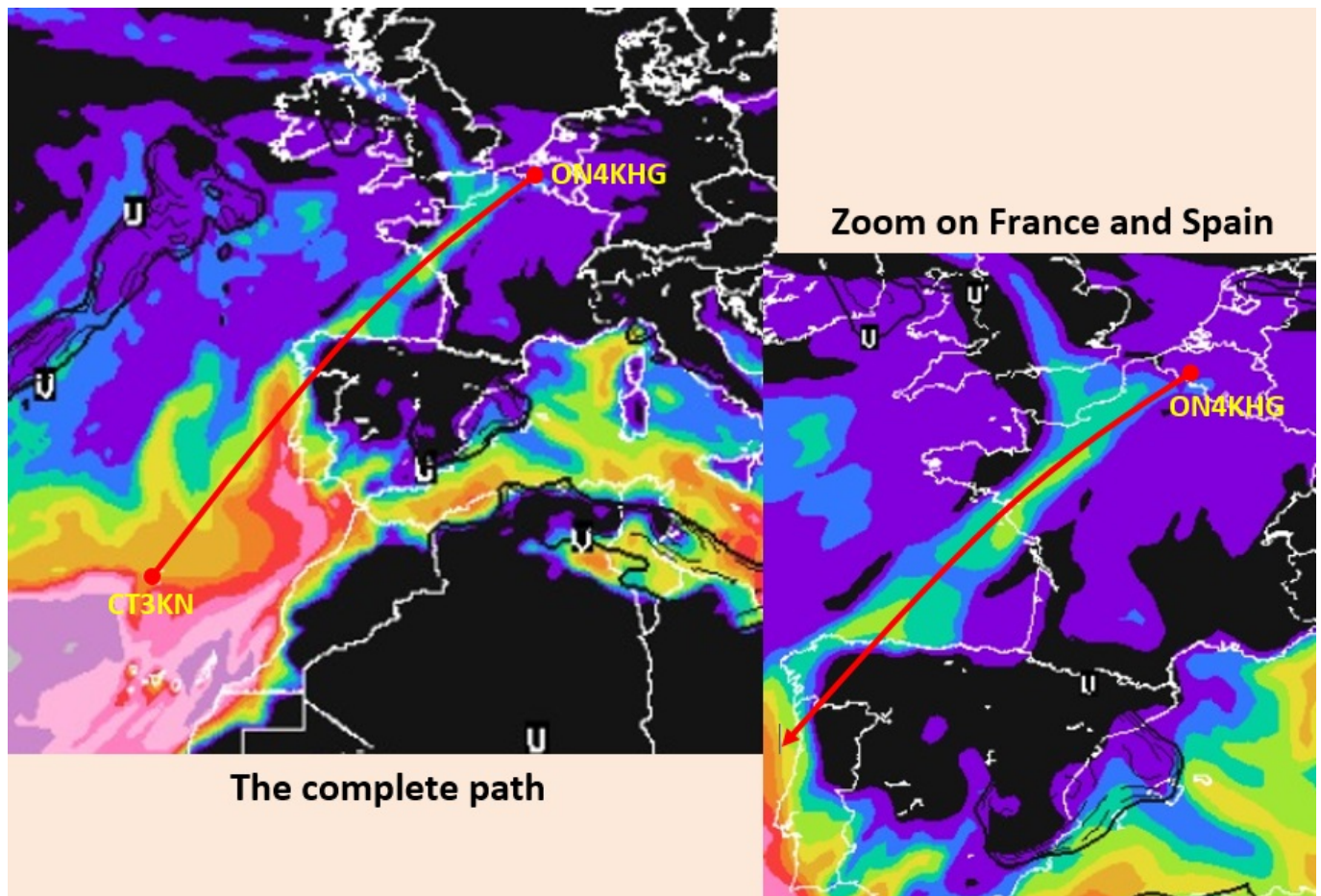
This is the complete path :



CT2HXM, Thierry sees "local" stations and my signal at 06:22 UTC. G0MBL (J001) is seen at 07:23 UTC. Surprisingly, no stations in between or around G0MBL and I. However, I checked PSK Reporter and there were not many F stations QRV on the path between 06:10 and 06:40 UTC (F0FWC and F0GFI, both in J010. F4KKV in IN98. F4ELJ in IN78). At 07:27 UTC, about when CT2HXM sees G0MBL, I have worked EB1B but CT2HXM doesn't see me then. So, the spot of 06:22 is probably MS, unless CT2HXM has turned his antenna between 07:23 and 07:27. Or, since I was in QSO with EB1B, not CQing, PSK reporter didn't report my signal around 07:23-27 UTC at CT2HXM' side ?



Looking at Hepbrun's Tropo maps, the duct was clearly visible. It seems it peaked (for my location) between 09:00 and 12:00 UTC. The map below shows the Tropo forecast at 09:00 UTC on August 19th :



This was a very thrilling experience, even though we couldn't complete a 2 ways QSO !

In such nice tropo conditions, working Ricardo (CT3KN) should be easy in Meteor-Scatter (actually "Tropo-enhanced MS"). Hopefully, as from IM12MT, Ricardo is located on the North side of Madeira, while Funchal (the capital city), is located on the South side, obstructed to Europe by mountains. I'm looking forward to try in MS with Ricardo, if not trying again and succeeding in Tropo !

Subregional VHF Contest July/Juillet 2021

Compte rendu du contest subregional VHF des 4 et 5 juillet 2020. Je n'ai été actif que partiellement, à savoir durant 9 heures sur les 24 que compte le contest. Je me suis attaché exclusivement à chercher les DX's (mode "S&P"). Une activité de contest "normale" semble enfin peu à peu reprendre en cette fin de période "covid".

Station utilisée : 2x9 él. DK7ZB et 1,2kW

Activity report of the subregional VHF contest of July 4th and 5th, 2020. I have been partly active, during 9 hours amongst the 24 the contests lasts. I exclusively focused on hunting DX's (mode "S&P"). Normal contest activity finally seems to be returning at the end of this "covid" period.

Station used : 2x9 el. DK7ZB and 1,2kW

QSO's : 73

Points : 43001

DXCC : 13 (G, HB9, OE, OK, DL, ON, PA, F, I, OZ, OM, EA, SP)

WWL : 40

Average km/QSO : 589

Top 10 DX QSO's :

EA2T	IN83FD	1005 km
OM2Y	JN88RS	990 km
OK1KTW	JN89IW	908 km
OL1B	J080IB	905 km
IW5BUX	JN54PF	897 km
F6KPH/P	JN03CB	885 km
OL7M	J080FG	885 km
OK1KHL	J080AC	858 km
OK1KCR	JN79VS	846 km
OL2J	JN79TI	844 km



Reception of 144 MHz beacons 0Y6BEC and IQ2MI in Tropo on July 23rd, 2021

Good Tropo conditions on Friday July 23rd, 2021 on 144 MHz.

Reception of the beacons IQ2MI in JN35WW (first time heard) and OY6BEC in IP62MD.



IQ2MI/B is the Europe highest VHF Beacon, located at "Capanna Margherita" refuge, at 4560 m asl. The power is only **500 mW** into a dipole !



Below, you can hear the recording of the beacon received on July 23rd, 2021. There is a transverter 144 > 28 MHz in front of the FT-857, so that the "real" frequency is 144.415 MHz, not 28.415.

Morse code translation :

V V V de IQ2MI/B JN35WW BAT 13.82V SUN 14.03V TIN 29C TOUT -32C www.arimi.it

Message format :

"V V V de IQ2MI/B JN35WW BAT nn.nnV SUN nn.nnV TIN xxC TOUT xxC www.arimi.it"

VBAT = battery voltage

VSUN = photovoltaic panel voltage

TIN = radio box temperature (°C)

TOUT = external temperature (°C)

You will hear that there is another beacon on the same frequency, it is DB0JW in J030.

Well, I have some doubts about the outside temperature... -32°C mid day during the summer (even if the altitude is high) ?

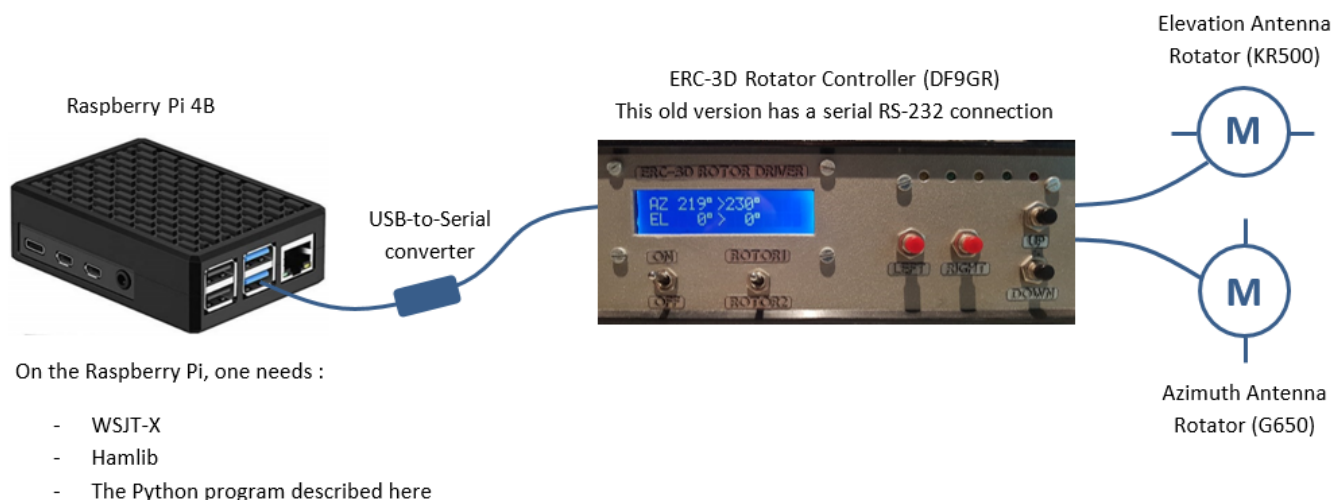
Now, heading north to the Faroes Islands, the 2m 0Y6BEC is located in that radome, in IP62MD :



[Antenna rotator control program for Raspberry Pi and WSJT-X](#)

For the digital modes traffic, I run WSJT-X on a Raspberry Pi (4B). So, no need to devote a “big” computer for that purpose, the Raspberry does the job perfectly well, and for less power consumption and noise. For my 144 MHz moonbounce operations, I wanted a simple system to drive my existing ERC-3D rotator controller (by DF9GR). That model is now obsolete but works flawlessly and I wanted to keep it as-is,

without changing anything. The ERC-3D is linked to azimuth (Yaesu G650) and elevation (Kenpro KR500) antenna rotators.



The search for a ready made program I did on the Internet was not successful, since I wanted to keep as-is my existing system. Having in mind that WSJT-X provides an `azel.dat` file including the coordinates of the moon, I decided to write a program to extract these data and to use them to command my ERC-3D, in first place to track the moon.

To be able to do it, a prerequisite is to have Hamlib installed on the Raspberry. What is hamlib : “the Ham Radio Control Library –**Hamlib**, for short– is a project to provide programs with a consistent Application Programming Interface (API) for controlling the myriad of radios and rotators available to amateur radio and communications users”. To date the current hamlib version is 4.2. and can be downloaded [here](#). How to install it on the Raspberry is out of the present scope, there are several tutorials on the Internet that clearly explain it (it is very easy).

Once Hamlib was installed, the program had to be written (it is available at the end of this post). It has been written in Python 3 and, since I’m not at all a programmer, it is my friend Didier (ON4KDV), who wrote the backbone of the program (many thanks to him !). I then customized it and added some features, a.o. the possibility to command the rotator controller not only to automatically track the moon but also to rotate/elevate the antennas wherever needed.

The program is named “Rotator Controller” and an icon placed on the desktop of the Raspberry allows to launch it easily. It opens in the command-line terminal.

Here is below how it works.

First you select the USB port to be used and then you have the choice :

- Manual rotation/elevation
- Automatic tracking of the moon, as from the azel.dat file out of WSJT-X
- Exit

We start first with the manual rotation/elevation. The possible options are self-explanatory. See also the code at the end, it provides many explanations (comments) too.

```
USB port to use (0-3) : 1
MANUAL rotation/elevation, type m
AUTOMATIC tracking, type a
EXIT, type e
m
ROTATE antenna (azimuth only) and set elevation to 0°, type r
ROTATE and ELEVATE antenna (azimuth and elevation), type l
Get current POSITION, type p
STOP, type s
EXIT, type e
```

When the option “r” is selected, if it was elevated before, the antenna will come back to the horizon (0° elevation) and towards the wished azimuth. “s” will stop immediately any rotation/elevation.

On the example here, we set the azimuth to 113°. Then we ask for the position (“p”) and we see that the antenna is effectively well on Az. 113°.

```

r
Rotate antenna to azimuth (°) : 113
set_pos: 113 0|RPRT 0

ROTATE antenna (azimuth only) and set elevation to 0°, type r
ROTATE and ELEVATE antenna (azimuth and elevation), type l
Get current POSITION, type p
STOP, type s
EXIT, type e

p
get_pos:|Azimuth: 113.00|Elevation: 0.00|RPRT 0

ROTATE antenna (azimuth only) and set elevation to 0°, type r
ROTATE and ELEVATE antenna (azimuth and elevation), type l
Get current POSITION, type p
STOP, type s
EXIT, type e

```

Then we select “l” and we set Az. to 113° and El. to 10°. Requesting the position by typing “p”, we get well 113° Az. and 10° El.

```

l
Rotate antenna to azimuth (°) : 113
Elevate antenna to elevation (°) : 10
set_pos: 113 10|RPRT 0

ROTATE antenna (azimuth only) and set elevation to 0°, type r
ROTATE and ELEVATE antenna (azimuth and elevation), type l
Get current POSITION, type p
STOP, type s
EXIT, type e

p
get_pos:|Azimuth: 113.00|Elevation: 10.00|RPRT 0

ROTATE antenna (azimuth only) and set elevation to 0°, type r
ROTATE and ELEVATE antenna (azimuth and elevation), type l
Get current POSITION, type p
STOP, type s
EXIT, type e

```

To come back to the main menu, you have to exit (“e”) and restart the program. Before restarting the program, we must start first WSJT-X, so that the azel.dat file is available. We then restart the program and we select a USB port not already in use by WSJT-X (for the CAT for example).

We select "a" for the automatic moon tracking. We are prompted to set a "refresh duration", it is the time interval (in minutes, but can be customized in seconds, see comments in the code) between two refreshes of the moon position (access to the azel.dat file in WSJT-X).

```
USB port to use (0-3) : 1

MANUAL rotation/elevation, type m
AUTOMATIC tracking, type a
EXIT, type e

a

Refresh duration (min) : 2
Moon azimuth (°) : 114.4
Moon elevation (°) : 3.1
set_pos: 114.4 3.1|RPRT 0

get_pos:|Azimuth: 113.00|Elevation: 4.00|RPRT 0

Waiting for the next update or CTRL+C to exit
```

Every x (2 in the example) minutes, as per defined by the "refresh duration", the moon position is derived from the azel.dat file, the antenna is rotated and elevated accordingly, and the actual antenna position (azimuth and elevation) is reported 5 seconds later.

```
Moon azimuth (°) : 115.3
Moon elevation (°) : 3.7
set_pos: 115.3 3.7|RPRT 0

get_pos:|Azimuth: 115.00|Elevation: 4.00|RPRT 0

Waiting for the next update or CTRL+C to exit

Moon azimuth (°) : 115.7
Moon elevation (°) : 4.0
set_pos: 115.7 4.0|RPRT 0

get_pos:|Azimuth: 115.00|Elevation: 4.00|RPRT 0

Waiting for the next update or CTRL+C to exit
```

If the moon is below the horizon, nothing happens and the program indicates "The moon is below horizon, antenna parked !", and so on until the moon rises above the horizon. To exit, press "CTRL" and "C".

```
a
Refresh duration (min) : 2
Moon azimuth (°) : 107.0
Moon elevation (°) : -2.3

The moon is below horizon, antenna parked !

get_pos:|Azimuth: 143.00|Elevation: 0.00|RPRT 0

Waiting for the next update or CTRL+C to exit

Moon azimuth (°) : 107.4
Moon elevation (°) : -2.0

The moon is below horizon, antenna parked !

get_pos:|Azimuth: 143.00|Elevation: 0.00|RPRT 0

Waiting for the next update or CTRL+C to exit
```

Here is the heavily commented python 3 program :

```
#!/usr/bin/python3
# coding: utf-8

# This script allows to rotate an antenna system either by manually typing in a
# heading (azimuth and/or elevation)
# you want the antenna to rotate to or by automatically tracking the moon azimuth
# and elevation.
# The Moon azimuth and elevation are extracted out of the file azel.dat provided by
# the WSJT-X suite.
# Beside WSJT-X, this script also makes use of the rotator controller daemon
# embedded in the Hamlib
# libraries. So, the prerequisites for this script to run are WSJT-X opened and
# running, and the
# Hamlib librairies installed on your Raspberry Pi.
# This script has been written in Python 3 by Didier, ON4KDV and Gaëtan, ON4KHG.
# My personnal system makes use of a Raspberry Pi 4B and an ERC-3D (DF9GR) antenna
# rotator controller.
# Both are linked with a USB (on the Raspberry Pi side) to RS232 serial (on the
# controller side) connection.
```

```

# First, import the time, system and subprocess functions needed in this script

import time
import os, sys
import subprocess

# Select the USB port of the Raspberry Pi onto which the rotator controller is
connected.
# Normally, as WSJT-X is started beforehand, the USB0 port is already assigned for
the CAT & PTT of WSJT-X.
# In my case, I usually select USB1, so, I type "1" below.

usb_port=input("USB port to use (0-3) : ")
print()

if usb_port == "0":

# The Rotctld(aemon) is opened with the command "rotctld -m 601 -r /dev/ttyUSB0 &".
# -m 601 is related to the GS-232A rotator protocol.
# You could have to have to select another number according to the protocol and
associated hardware in use on your side.
# The list of supported protocols can be obtained by typing "rotctld -l" in the
terminal.
# ttyUSB0 is the serial port (USB0) of the Raspberry onto which the rotator
controller is connected.

        subprocess.call('rotctld -m 601 -r /dev/ttyUSB0 &', shell=True)

# And so on for the other USB ports.

elif usb_port == "1":

        subprocess.call('rotctld -m 601 -r /dev/ttyUSB1 &', shell=True)

elif usb_port == "2":

        subprocess.call('rotctld -m 601 -r /dev/ttyUSB2 &', shell=True)

elif usb_port == "3":

```

```

subprocess.call('rotctld -m 601 -r /dev/ttyUSB3 &', shell=True)

# Select the working mode : manual, automatic (moon tracking) or exit.

Mode=(input("MANUAL rotation/elevation, type m\nAUTOMATIC tracking, type a\nEXIT,
type e\n\n"))

if Mode == "m":

# If "m" is chosen, the manual rotation mode is selected.

while (True):

# Then, select what you want to do : rotate/elevate the antenna, get its current
position, stop the rotation in case
# of emergency or exit.

Submode=(input("\nROTATE antenna (azimuth only) and set elevation to 0°, type
r\nROTATE and ELEVATE antenna (azimuth and elevation), type l\nGet current POSITION,
type p\nSTOP, type s\nEXIT, type e\n\n"))

if Submode == "r":

# If "r" is selected, instruct towards which azimuth the antenna has to rotate. With
this selection, if the antenna was previously elevated,
# it will be set back to 0° (no elevation) too.

az=(input("\nRotate antenna to azimuth (°) : "))
az=az.strip()

# Build and execute the command "set_pos". The default port used by the
rotctld(aemon) is 4533.

command_az='echo "|set_pos ' + az + ' 0" | nc -w 1 localhost 4533'
os.system(command_az)

elif Submode == "l":

```

If "l" is selected, you can set whatever azimuth and elevation angle.

```
az=(input("\nRotate antenna to azimuth (°) : "))
```

```
az=az.strip()
```

```
el=(input("Elevate antenna to elevation (°) : "))
```

```
el=el.strip()
```

```
command_azel='echo "\set_pos ' + az + ' ' + el + '" | nc -w 1 localhost  
4533'
```

```
os.system(command_azel)
```

If "p" to get the position of the antennae is selected, build and execute the command "get_pos".

```
elif Submode == "p":
```

```
command_azpos='echo "\get_pos" | nc -w 1 localhost 4533'
```

```
os.system(command_azpos)
```

If "s" to (emergency) stop is selected, build and execute the command "S" (Stop).

```
elif Submode == "s":
```

```
command_stop='echo S | nc -w 1 localhost 4533'
```

```
os.system(command_stop)
```

```
else:
```

If none of the above is selected, then exit the script.

```
exit()
```

```
elif Mode == "a":
```

If "a" is chosen, the automatic (moon tracking) mode is selected.

Set the refresh rate in minutes of the moon azimuth and elevation.

With narrow beamwidth antennas (microwaves), the refresh rate has to be fast.

With wide beamwidth antennas, the refresh rate can be slower. On 2m, I chose 4 minutes.

If you want a refresh time in seconds, remove the line "ref=ref*60" and rename "min" into "sec"

in the line ref=int(input("\nRefresh duration (min) : ")).

```
ref=int(input("\nRefresh duration (min) : "))
ref=ref*60
```

```
while (True):
```

Open the file azel.dat of WSJT-X in read mode. Make sure the path "/home/pi/.local/share/WSJT-X/azel.dat" is the same

than the path defined in the settings of WSJT-X (see the WSJT-X user guide).

The first line of the file is read and placed in the variable "txt" and then the file is closed.

```
f=open("/home/pi/.local/share/WSJT-X/azel.dat","r")
txt=f.readline()
f.close()
```

```
if ("Moon" in txt):                # Check that the first line contains the
word "Moon".
```

```
    p=txt.find(",")                # Search for the 1st comma (,).
```

```
    if (p > 1):                    # If the 1st comma has been found
```

```
        txt=txt[p+1:]              # what is in front of the 1st comma is
removed, including the comma.
```

```
        p=txt.find(",")            # Search for the 2nd comma.
```

```
        if (p > 1):                # If the 2nd comma has been found
```

```
            az=txt[0:p]             # the text in between the 1st and 2nd
commas is saved in "az" (azimuth).
```

```
            az=az.strip()           # Spaces before and after are removed, only
the figures/sign are kept.
```

```
            print("Moon azimuth (°) :",az)  # The moon azimuth is displayed.
```

```
            txt=txt[p+1:]           # What is in front of the 2nd comma is
removed, including the comma.
```

```
            p=txt.find(",")          # Search for the 3rd comma.
```

```
            if (p > 1):              # If the 3rd comma has been found
```

```

        el=txt[0:p]          # the text in between the 2nd and 3rd
commas is saved in "el" (elevation).

        el=el.strip()        # Spaces before and after are removed, only
the figures/sign are kept.

        print("Moon elevation (°) :",el)  # The moon elevation is
displayed.

# Build the command "set_pos" as from the az and el variables extracted above.

        command_azel='echo "|\\set_pos ' + az + ' ' + el + '" | nc -w 1
localhost 4533'

# Convert az and el from text to float (numerical) format.

        f_az=float(az)
        f_el=float(el)

# Execute the command only if the azimuth is between 0 and 360° and if the elevation
is between 0 and 90°.

        if(f_az >0 and f_az <361 and f_el >0 and f_el <91):
            os.system(command_azel)

# Otherwise the moon is below the horizon and nothing is executed but displaying
that the moon is below horizon.

        else:
            print('\\nThe moon is below horizon, antenna parked !')

# We wait 5 seconds, the time for the antenna to rotate (it may actually take
longer, depending on the position of
# the antenna at startup of the tracking) and then we display the position of the
antenna.
# Finally, we indicate that we wait until the next update, according to the refresh
duration (ref) defined above.

        time.sleep(5)
        print()
        command_azelpos='echo "|\\get_pos" | nc -w 1 localhost 4533'

```

```
os.system(command_azelpos)
print('\nWaiting for the next update or CTRL+C to exit\n')

time.sleep(ref)

# If none of the working mode (manual or automatic moon tracking) was selected, exit
and close the script.

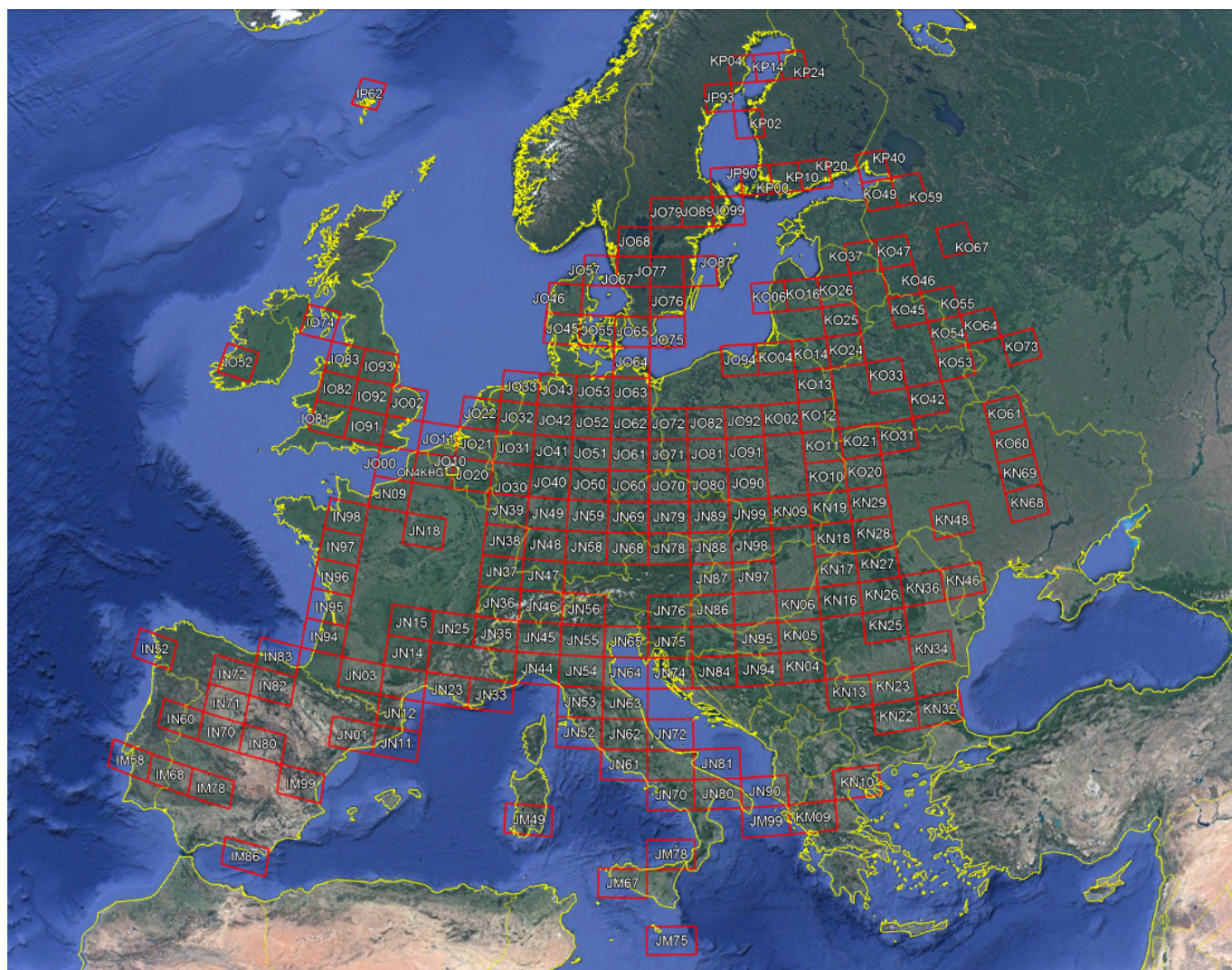
else:
    exit()
```

Geminids / G minides 2020

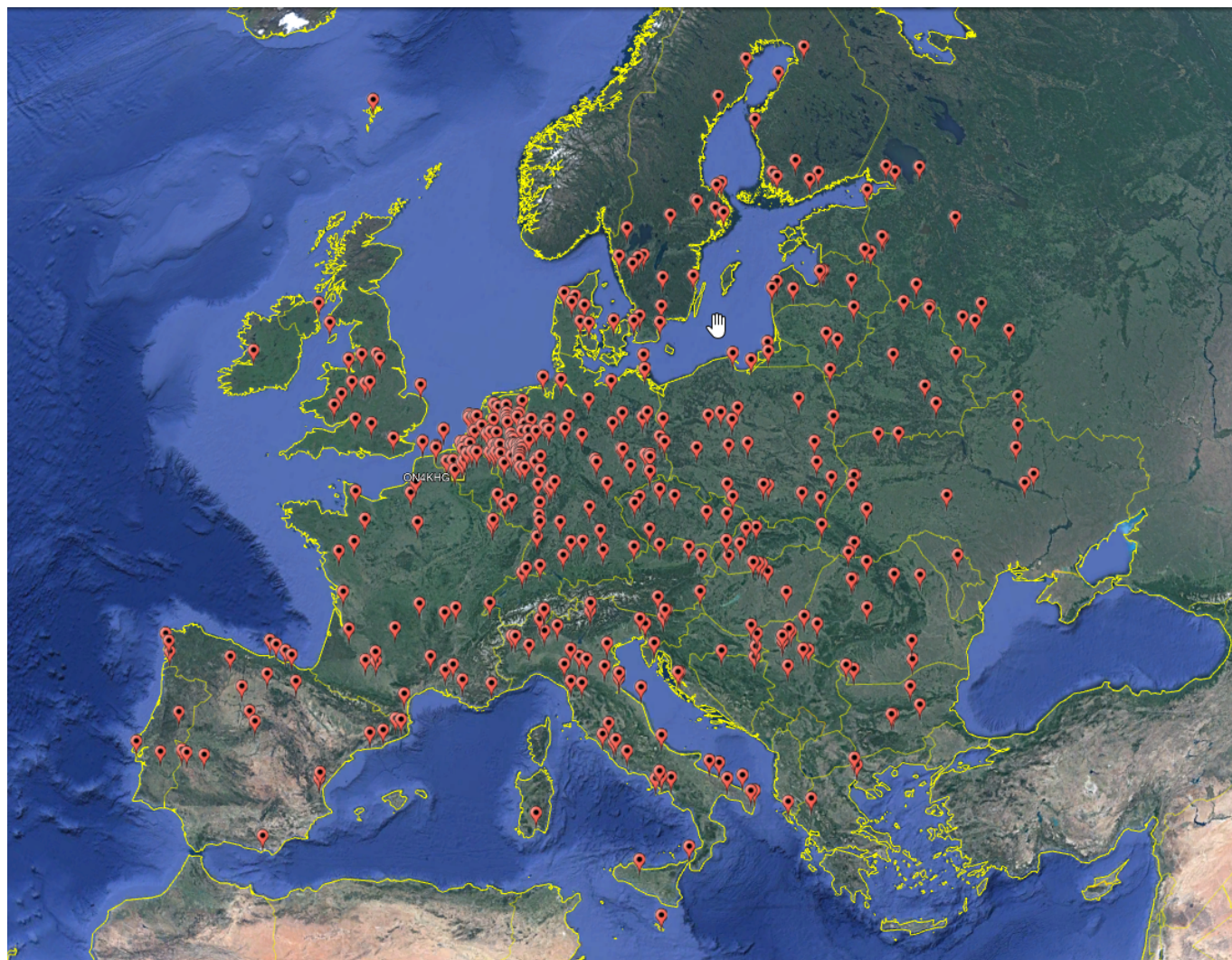
Ci-dessous quelques cartes des stations entendues durant les G minides 2020 sur 144 MHz, des stations qui m'ont entendu et des stations que j'ai contact es.

Below are some maps of the stations heard during the 2020 Geminids on 144 MHz, the stations that heard me and the stations that I worked.

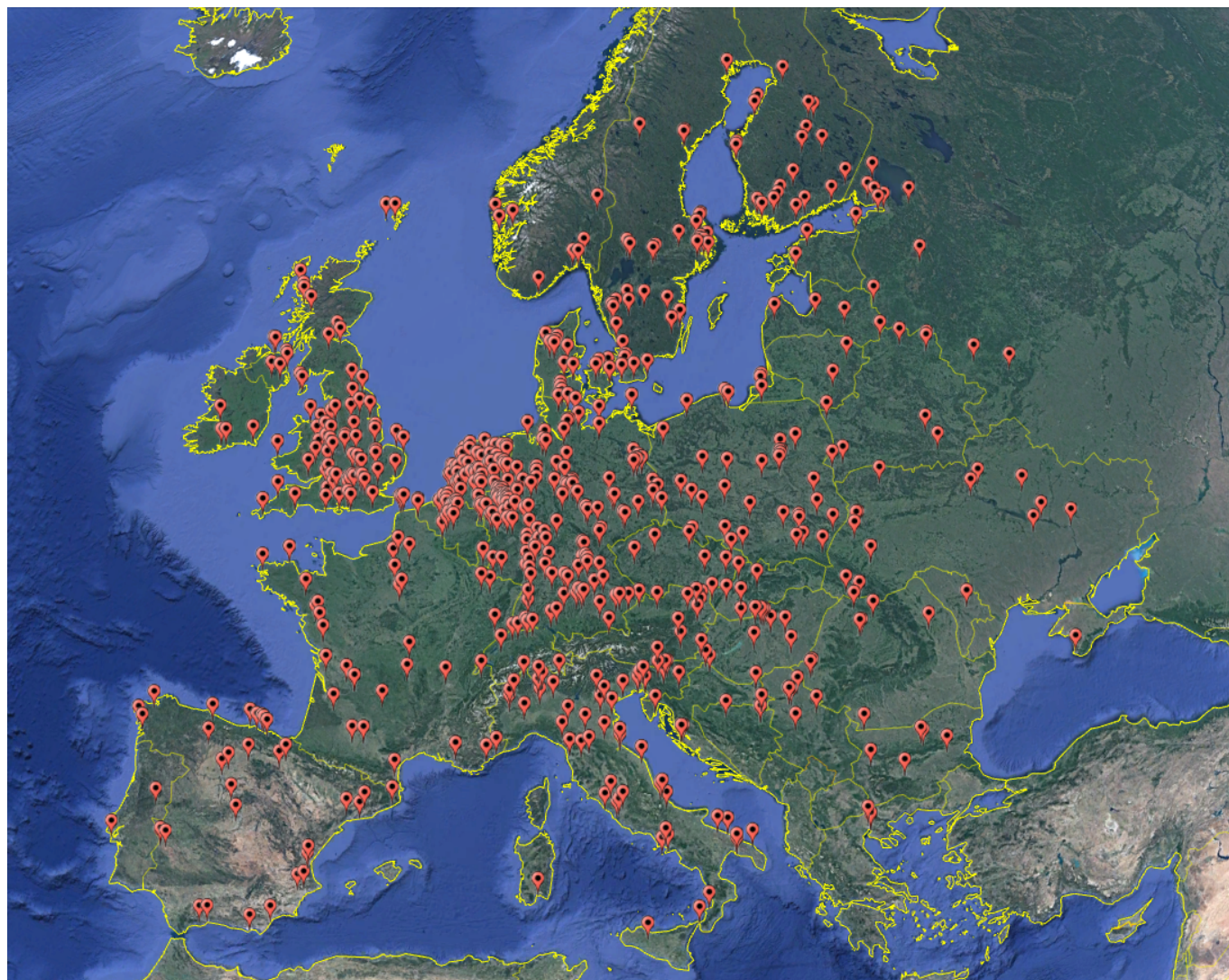
QRA locators entendus (Tropo ou MS) / [QRA locators heard \(Tropo or MS\)](#) :



Stations entendues (Tropo ou MS) / Stations heard (Tropo or MS) :



QRA locators où j'ai été entendu (Tropo ou MS) / [QRA locators where I have been heard \(Tropo or MS\)](#) :



Stations contactées (MS) / [Stations worked \(MS\)](#) :

