

# ECHOES

## METEOR SCATTER WITH RTL-SDR

VERSION 0.26 - MAY 31, 2019



Copyright © Giuseppe Massimo Bertani, 2019  
This work is licensed under the Creative Commons  
Attribution-ShareAlike 4.0 International License.

To view a copy of this license, visit  
<http://creativecommons.org/licenses/by-sa/4.0/>  
or send a letter to Creative Commons,  
PO Box 1866, Mountain View, CA 94042, USA.

## Contents

1 Introduction.....	3
2 Acronyms.....	4
3 References.....	5
4 Overview.....	6
4.1. External dependencies.....	6
4.2. Core .....	7
5 Installation.....	9
5.1. Windows platform.....	9
5.2. Linux platform.....	9
5.3. Raspberry-PI.....	10
6 Basic usage.....	11
6.1. Launching from the shell.....	11
6.2. Launching from desktop.....	13
6.3. The Main Window.....	14
6.3.1 Pushbuttons.....	14
6.3.2 Device settings.....	14
6.3.3 FFT settings .....	15
6.3.4 Output settings.....	16
6.3.5 Report settings.....	18
6.3.6 Preferences.....	19
6.3.7 Status bar.....	20
6.4. The Waterfall window.....	21
6.4.1 Graphic panes.....	21
6.4.2 Waterfall controls.....	22
6.4.3 Graphs controls.....	23
6.4.4 Notch filters .....	23
7 Playing with Echoes .....	25
7.1. Operating modes.....	25

7.1.1Continuous recording.....	25
7.1.2Periodic recording.....	25
7.1.3Automatic snapshot recording.....	25
7.2.From FFT output points to pixels.....	27
7.3.Examples .....	28
7.4.Files contents and naming conventions.....	33
7.4.1Configuration files .....	33
7.4.2Log file.....	33
7.4.3Screenshots.....	33
7.4.4Statistic CSV table file.....	34
7.4.5Plot files.....	36
7.4.6Daily report.....	39
7.4.7Full report.....	40
7.5.Daily archive.....	40
7.6.Test patterns.....	42

# 1 Introduction

Echoes is a radio spectral analysis software for RTL-SDR devices, designed for meteor scattering purposes.

RTL-SDR are very cheap Software Defined Radios that use DVB-T TV tuner dongles based on Realtek's RTL2832U chipset. With the combined efforts of Antti Palosaari, Eric Fry and Osmocom it was found that the signal I/Q data could be accessed directly, which allowed the DVB-T TV tuner to be converted into a wideband software defined radio by the means of a special software driver.

This driver must be installed in alternative to the official driver shipped generally in a CD-ROM along the dongle, if not already installed in your OS in the factory.

*Echoes* doesn't demodulate neither decode any human-made signal. Its main goal is to analyze and record the total power of natural signals and generate screenshots and tabular data (CSV, GNUplot) output in presence of particular peaks in a selected narrow range of frequencies. Since there is no demodulation, there is no provision for audio listening, except for a notify sound when an event has been recorded.

*Echoes* expresses the output power as dBfs – decibels at full scale – since the output signal from RTLSDR is not calibrated. The zero dBfs value means the maximum amplitude of that signal. Lower values give negative dBfs, down to about -150 depending on your hardware.

A conversion to dBm requires to multiply the dBfs by a conversion factor depending of your hardware and the working frequency chosen.

## 2 Acronyms

Acronym	Meaning
DVB-T	Digital video tuner
RTL	Realtek: producer of RTL-8139 receiver chip.
SDR	Software Defined Radio
UI	User Interface
GUI	Graphical User Interface
GRAVES	<i>Grand Réseau Adapté à la Veille Spatiale</i> , radar station located in Dijon (F) aimed to space debris monitoring.
RPM	Redhat Package Manager
OM	“Old Man” radio amateur
OSS	Open Source Software
RF	Radio Frequency
FFT	Fast Fourier Transform

### 3 References

1. RTL-SDR open source library:  
<http://www.rtl-sdr.com>
2. List of devices supported by RTL-SDR. Some of them are obsolete and no more available:  
<http://sdr.osmocom.org/trac/wiki/rtl-sdr>
3. FFTW – the fastest FFT library of the World – is used in *Echoes* to produce the spectrogram.  
See <http://www.fftw.org>
4. GPUFFTW – FFT for Raspberry PI's GPU (future)
5. Echoes has been developed in C++ language, under Qt5 framework. <http://www.qt.org>
6. IMO meteor scatter theory <https://www.imo.net/observations/methods/radio-observation/reflection/>.
7. IMO practical meteor radio observations <https://www.imo.net/observations/methods/radio-observation/practical/>
8. GRAVES radar descriptions can be found on Wikipedia or here:  
<http://f6crp.pagesperso-orange.fr/ba/graves.htm> (in french, more accurate)
9. Zadig WinUSB drivers: <http://zadig.akeo.ie/>
10. GNUplot site: <http://www.gnuplot.info/>
11. I/Q data: <http://whiteboard.ping.se/SDR/IQ>
12. dBfs vs. dBfs: <http://dsp.stackexchange.com/questions/19615/converting-raw-i-q-to-db>
13. Arch Linux ARM Wiki: <https://archlinuxarm.org/wiki>
14. Nongles N3 RTL-SDR receiver <http://www.nongles.com/detail-n3-receiver.html>
15. Gruppo Astrofili della Bassa Bergamasca <http://www.gabb.it>
16. [Using qirx SDR and DAB signals to calibrate RTL-SDR dongles](#)

## 4 Overview

### 4.1. External dependencies

This manual is about what the yellow box at the bottom of Fig. 1 does. The remaining boxes are the hardware and software components that *Echoes* needs to do its job, shortly described below. More informations about these components can be found following the reference links (par.3), but it's not mandatory to know how they work to do interesting things with the program.

#### ECHOES DEPENDENCIES

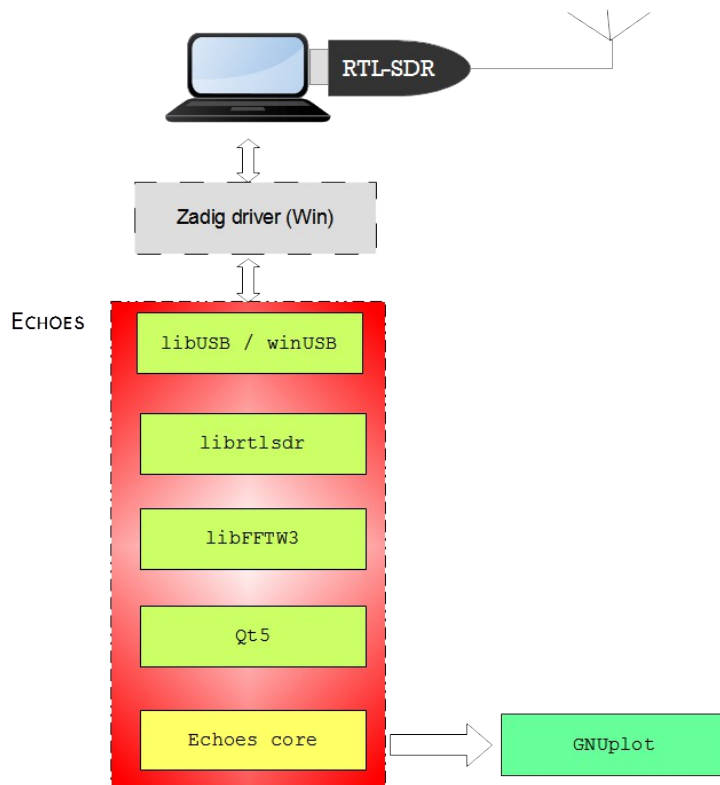


Fig. 1: Software and hardware components needed to run Echoes

The Zadig tool is needed only once to install the libUSB/winUSB drivers. If you already used HDSDR on the same machine, they are probably already present. These drivers are needed to access some functions of USB ports that normally are managed by the operating system only. Please note that the original DVB-T driver supplied with the dongles should never be installed if you use the dongle as RTL-SDR.

librtlsdr is an open source shared library available for Windows and Linux. It is deployed along with some textual utilities to play with your dongle (rtl\_fm for instance is a simple radio receiver).

libfftw3 implements the *Fastest FFT in the World*. Being a simple user, I take this statement for true... but I'm still open to experiment with better OSS if exists, let me know.

**Qt5** is the fifth version of the multiplatform GUI libraries and related tools by Trolltech (then Nokia, now M\$). *Echoes* uses only the open source licensed libraries and tools.

Finally, **GNUPlot** is an external program for data plotting. It is not mandatory to install it to run *Echoes* but is useful for post-processing.

## 4.2. Core

Let's see now what is inside the yellow box in Fig. 1 . The Fig. 2 resumes its functionalities in a block scheme:

### ECHOES CORE

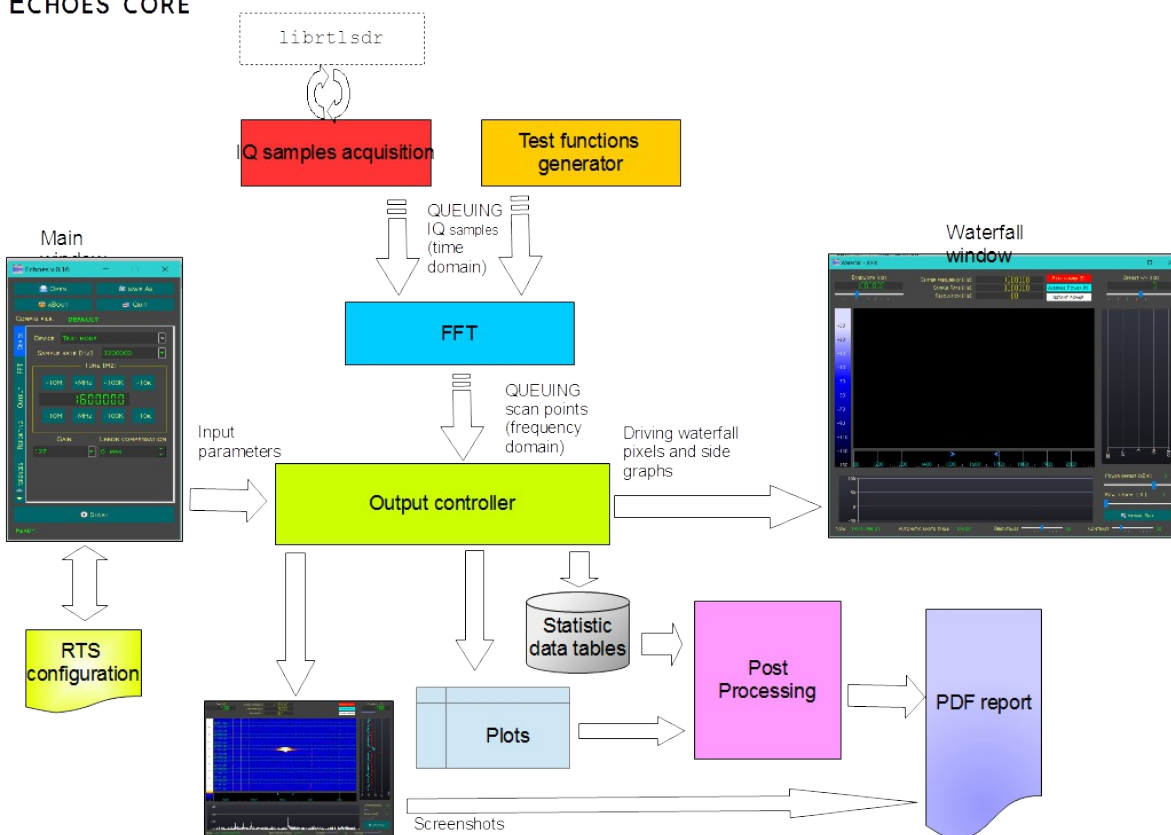


Fig. 2: *Echoes* functional blocks scheme

Once set the desired parameters in main window, by pressing the Start button the output controller (green block) starts the radio acquisition thread (red block) that fetches at full speed the radio signal samples.

If the dongle is not present, the controller starts the test functions generator thread (yellow block) instead. It generates fake signal samples to simulate the graphical output and data files generation.

The samples in time domain are then forwarded to the FFT algorithm (cyan block) . The output is a sequence of values that specify the power of the input signal in frequency domain. The number of

these values depends of the sample rate and the resolution chosen. Each sequence (*scan*) produced by the algorithm, after some processing, is sent to the waterfall and contributes in generating plots and statistic data tables.

These data files are then post-processed (pink block), after acquisition stop, to generate a report (gray block). The settings changed in the windows are saved in a default configuration file (yellow/white block) that is automatically reloaded at next program's start. These settings can be saved with different names and retrieved later as needed.



## 5 Installation

The binaries of version 0.26 are actually available as Windows 32/64bit installer (.exe) , RPM binary packages for x86\_64 and Raspberry-PI (Arch distro).

### 5.1. Windows platform

The Windows installer (*Install-echoes-0.26-Win32.exe* , *Install-echoes-0.26-Win64.exe*) have been tested under Win7 and Win10.

A prerequisite for installation is the presence of *WinUSB.dll* or similar driver library. Such libraries are available at Zadig <http://zadig.akeo.ie/> into a nice installer.

So when Zadig installer asks which dll should be installed, the best choice is *WinUSB.dll*.

When the *Echoes* installer finds an older version of the program on the computer, it prompts the user for automatic removal. This prompt in fact does not take in count the program version, so a downgrade is still possible in case the newer version doesn't work as expected.

After the old package removal, the installation of the new one proceeds. This process produces a log file *echoes-install.log* that is saved in the documents directory, to be analyzed in case of install / uninstall failures.

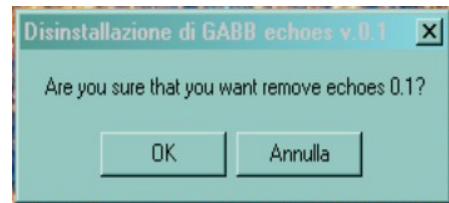


Fig. 3: Prompt for removal of an already installed Echoes version

### 5.2. Linux platform

Under Linux, RPM package files for x86\_64 have been generated. The reference distro is OpenSuse Leap42 but they should work even for other RPM-based distributions when the dependencies are satisfied.

The dependencies of *Echoes* are :

- *rtl-sdr* library
- *rtl-sdr-devel* (for development purposes only)
- *fftw* (version 3 or 2)
- *Qt* (version 5) including *QtCharts*
- *pulseaudio* or *sox*
- *GNUplot* (optional)

different distros could call these binary packages with somewhat different names, causing the install to fail. If you know that the above packages have been regularly installed on your Linux box, before giving up, try a

```
rpm -i echoes-0.26-0.????.rpm --force --nodeps
```

that will force the installation of the package, regardless its dependencies. In this way you can still check if the program works and - if not - you can still remove it later.

### 5.3. Raspberry-PI

Due to the migration to Qt5, my Raspberry PI B+ (1<sup>st</sup> generation) has been upgraded to *Arch Linux* distribution, being nowadays the unique distro including both Qt5 and QtCharts packages.

*Echoes* runs, but too slowly to be useful for meteor scatter, a scan line takes about 400ms to appear on screen. For this reason, A full console UI can be chosen at startup (command line switch `-c`, see 6.1. ) to allow running *Echoes* without X server (*Xorg*), saving some memory. In this way, no screenshots can be taken, but spectra can be saved in *GNUplot* format to be displayed later.

On a RPI this means more free RAM, since *Xorg* is no more needed, and – hopefully - more speed, since there is no need to update a waterfall diagram.

The RPI B+ 3<sup>rd</sup> generation is effectively faster and thanks to 1GB of RAM is able to run *Xorg*, allowing to run *Echoes* with graphics. Such power needs to be cooled by adding heatsinks, better if cooled with a little fan.

Unfortunately, the spectra produced are really noisy, showing vertical stripes at regular distance. I tried to change power source and stop the fan, but the problem remains, at least for me. In fact, I know about another *Echoes* user, using the same RPI, having no such problems. The first generation RPI didn't have.

So actually I don't know if it's a specific problem of my board or something wrong in my setup.

## 6 Basic usage

### 6.1. Launching from the shell

The RTL-SDR dongle must be plugged in before invoking the program because it won't be recognized later.

Typing

```
echoes -h <Enter>
```

from the command prompt, the following options list is displayed. Under Windows, you should launch the program from the directory where the executable resides, since the install procedure does not insert *echoes* into the system path.

```
RF spectrum analyzer for RTL-SDR devices v.0.12
Architecture: x86_64 ABI: x86_64-little_endian-llp64
(C)Giuseppe Massimo Bertani 2016.

Usage : echoes [options]

Options:
-?, -h, --help    Displays this help.
-l <language>     loads the language qm file specified (defaults to local
                  language, otherwise english).
-s <config>       loads the settings from user config file given.
-w <wdName>       sets this directory as working directory instead of
                  $HOME/.echoes
-n <level>        log level:
                  0: do not create a program log
                  1: only fatal messages (crashes) are logged
                  2: logs fatal and critical messages (alerts about possible
                     crashes)
                  3: logs warnings too (the program does not behave as expected)
                  4: logs status messages too (useful for console mode).
                  5: logs everything including debug messages (huge logs!).

-c               console mode: acquisition starts automatically, no windows
                  will be shown.
-r               restores the hardcoded default settings (config_file if given
                  will be ignored).
-v               verbose debug output on text console.
```

Fig. 4: Command line switches

For normal usage, none of these options are needed; *Echoes* can be started without arguments simply by clicking on its icon and any further setting can be performed through its GUI, but there could be particular situations where these options can be useful.

- The **-c** switch starts *echoes* without a GUI (console UI) . The settings used are the default ones, unless a configuration file is specified with parameter **-s<config file>**. It can be launched from an **xterm** (Linux) or command window **cmd.exe** (Windows) and the acquisition starts immediately. This working mode is suited for headless, stand-alone stations and older Raspberry-PI boxes. Being no GUI, no snapshots will be produced; but if GNUplot output is enabled, each event can be captured in a color-mapped graph, that can be

displayed later with GNUplot, looking similar to screenshots.

The configuration files can be prepared and tested with the GUI, to be later opened in console UI.

**Linux only:** Starting Echoes as `/usr/bin/echoes -c ....` from an *xterm*, the program runs entirely into that window, without displaying any further. This matches the behavior under Windows, when you start *echoes -c* from the command window *cmd.exe* but in fact, you're still running Echoes under a graphical environment.

Under Linux nevertheless, you can start your system in multiuser mode without graphics. You can set that mode with the command `systemctl set-default multi-user.target`. By appending the command `/usr/bin/console_echoes -c ...` at the end of the file `/etc/rc.d/boot.local`, Echoes will be started automatically as a service at next reboot, without needing user login. Its textual output can be watched by the means of the command `journalctl -f | grep echoes`.

*console-echoes* is simply an alias for *echoes*. When called with that name, Echoes recognizes you want to exclude totally the graphical support.

- The GUI language is chosen automatically depending of the operating system localization settings. Actually, the only alternative to the hardcoded language (English) is Italian, so the italian language will be displayed automatically on italian PCs. If you desire to override the default behavior and load a precise translation file it can be specified with the option `-l`. This option is useful to test new translations without touching the program directories and without administration privileges.
- *Echoes* by default produces a lot of debugging text output that can be read in the *echoes.log* file, under the default working directory. This file is re-created from scratch each time *Echoes* is launched. By setting the `-n 0` option, the creation of this file will be inhibited, saving some CPU time and disk resources. Higher numbers (up to 5, with default 3) give more output.
- The GUI settings can be saved in user defined rts files. This can be done by pressing the “Save” pushbutton and specifying a path and a file name. The rts extension will be added by default. These files can be reloaded in future sessions by specifying the `-s` option followed by a *rts* file path.
- By default, each time it starts, *echoes* looks for a default configuration file in the default working directory called *default.rts*. This file is updated with the actual parameters set through the GUI each time the acquisition starts or when the program terminates regularly by pressing the “Exit” pushbutton. In this way, the settings of the program remains persistent between consecutive invocations. The `-r` option can be set when you desire to reset the parameters to the hard coded settings (`-s` will be ignored in this case)
- The debug messages dumped in *echoes.log* can be watched in real time by specifying the `-v` option. In this way, the message will be dumped on *stderr* too. (Under Windows, this means that they will appear in the command prompt window that opens automatically with the program's ones.)
- The default working directory (*echoes/* or *.echoes/* is normally created under the user's home

directory) but another working directory can be specified thru the `-w` option. In this directory will be stored the files generated by the program (logs, screenshots, data files, configurations, reports). This option, like `-c`, has been designed for headless stations to allow saving data on removable hard disks.

## 6.2. Launching from desktop

However, *Echoes* can be launched also by clicking on its desktop icon. In this case the program is launched without command line arguments, using the default parameters. Under Windows platform, the application pops up the three windows showed in Fig. 5

1. The “Main window” (right) contains the controls related to tuning, acquisition, data recording.
2. The “Waterfall window” (center) contains the real-time graphical output of the program: a FFT waterfall with colors scale and tuning ruler, an instantaneous FFT output graph under it and a total power vs. time graph at its right.
3. Last is the “Console window” (left) that opens automatically under Windows platforms to display some status messages and warnings generated by *rtl-sdr* driver. Moreover, this window can display also the program log (*echoes.log*) when *Echoes* is started with `-v` option. Under Linux, the same behavior is achieved by launching the program from a terminal shell session.

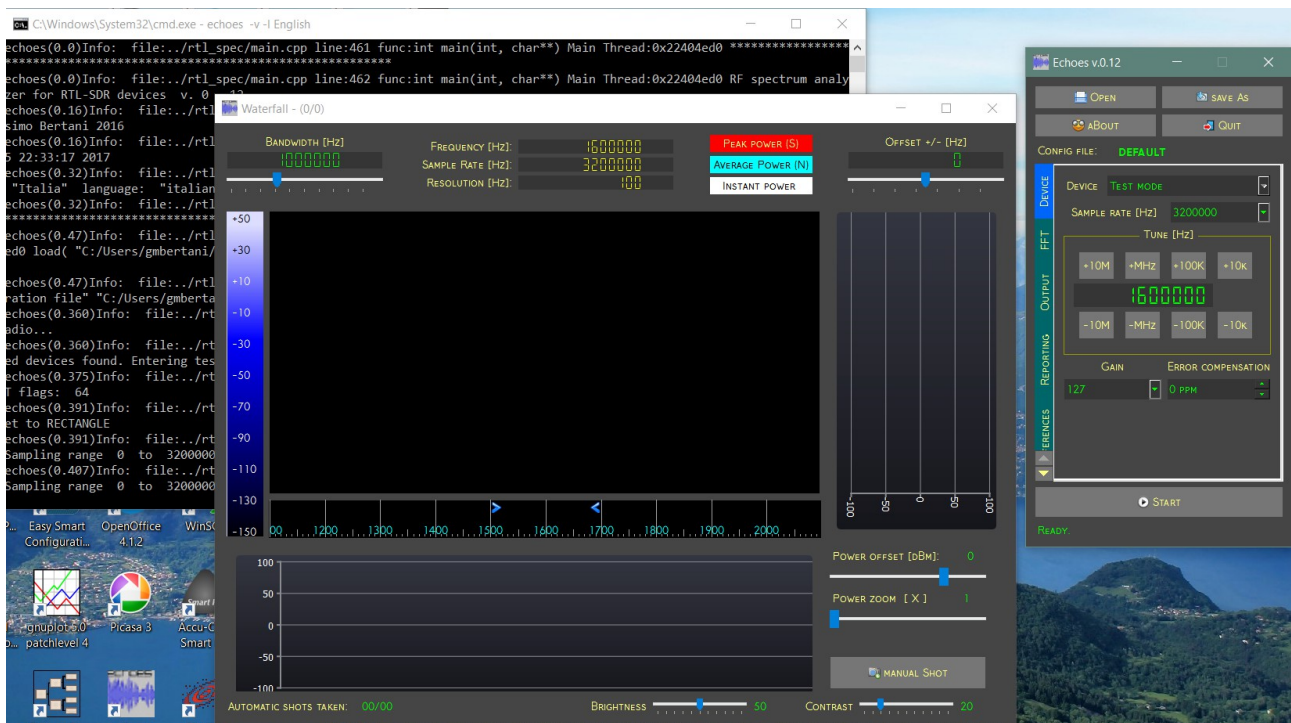


Fig. 5: *Echoes* launched on Windows10

### 6.3. The Main Window

This window controls the data acquisition and general program behavior. It's a small one, subdivided in tabs. The variable fields in window have green color, while fixed texts and pushbuttons are yellow.

After program startup (unless a configuration file has been chosen at command prompt via the switch `-s`) the program configures itself following the settings present in *default.rts*. The first time the program is invoked - when that file does not exists yet - it takes the hardcoded (“factory”) settings.

Each time the acquisition starts, all the settings are implicitly recorded in that file, including user's changes. The user can save the settings in dedicated files to be retrieved later as needed by the means of “Save” pushbutton.

Some installation-dependent parameters previously kept in *rts* files, like the windows geometries, the gnuplot path and the events count, have been moved to a local configuration file (The registry under Windows, *.config* files under Linux) in order to allow the use of *rts* files produced by other stations without impacts on these local settings.

#### 6.3.1 Pushbuttons

The two pushbuttons on the top “Open” and “SaveAs” manage the user's configuration files (*.rts*). Immediately below them, are the “About” and “Quit” pushbuttons. The first opens a dialog with program version and other general informations, the second is self-explanatory. Note that there is no a simple “Save” button; this operation can be done by pressing “SaveAs” then press “Ok”, since the proposed file name is the name of the last configuration file opened. The active configuration name is showed just below the buttons.

The last pushbutton is “Start/Stop” at the bottom that starts the data acquisition thread. Below it, the status bar shows the acquisition status and error messages. While acquisition is active, most of main window's controls are locked except for the pushbuttons for acquisition Stop and Quit. The controls on the Waterfall instead are always available.

#### 6.3.2 Device settings

The “Device” tab allows to control the device parameters. When acquisition is started, the main window doesn't allow to change the active settings, but it's still possible to watch them and browse the tabs. When one or more dongles are connected at the USB, the name of the selected one is displayed under the “Device” control. Clicking on it, it's possible to choose the desired dongle for acquisition. That name can be saved in configuration file so when it will be loaded again, the same dongle will be opened. Configuration files can be saved dedicated to each dongle, and this attribution will work until the dongles will be removed or changed the USB socket they are plugged in.

The “Sample Rate” control shows the actual sample rate (SR) setting. The available rates are stored in the device driver and can be shown by clicking the right arrow, that allows to select a new value.

Below the sample rate, there are the tuner frequency controls, a LCD-like display that shows the

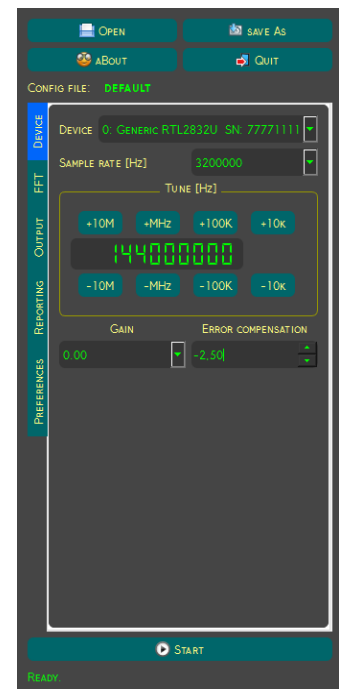


Fig. 6: main window playing test patterns, Device tab.

frequency in Hz and some pushbuttons to change this frequency. By right clicking on the LCD, a dialog box allows to set a precise frequency by keyboard.

The “Gain” control behaves like the “Sample Rate”: the available gains depends on the device selected and can be browsed by pressing the right arrow button. Please note that *Echoes* always disables the AGC; unluckily it's not possible to exclude it completely via software only.

“Error PPM” is used to enter a fixed value to compensate the tuner PPM error. You should try, by the means of a transceiver, to generate a carrier at a precise frequency – let's say 144.000.000 hz and expect the waterfall showing a vertical line near the same frequency; then play with “Error PPM” until the vertical line reach the waterfall's center. As alternative, please read the article [16] .

The error compensation value can be specified with a 2 decimals precision. The integer part is managed by hardware, while the fractional part is used to apply a transparent offset correction to the waterfall for a more precise centering.

Remember to write down the PPM value on a sticky label to attach on your dongle. It will be stored in configuration file anyway.

### 6.3.3 FFT settings

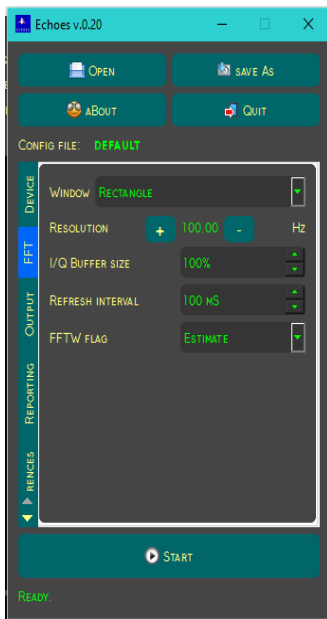


Fig. 7: The FFT tab

The second tab “FFT” encloses the controls that have effect on the spectrogram quality.

The “Window” control applies an additional algorithm to the FFT output that allows a better separation between the spectrogram output points and reduce the noise level (N). There are five possible selectable algorithms, while the default “Rectangle” is the flattest one.

The selectable “Resolution” step values are mostly non-integer values since they depend of the sample rate chosen. For performance reasons, the FFT transforms the input signal to a number of frequency “points” that must be a power of two, so the available resolutions have precise values depending of the sample rate.

The resulting number of points can be obtained by dividing the sample rate by the resolution step and the result must be a power of two.

The smallest is the resolution step, the lower is the noise level (N) and the higher is the time to spend to get raw data from the dongle.

The I/Q samples buffer size can be adjusted as percentage of the sample rate, to select sizes smaller than 100% (the hardcoded default value).

Smaller buffer sizes makes the acquisition thread to run faster, unfortunately this increases also the rate of FFT calls and decreases the quality of the waterfall. For this reason, for most cases, there won't be the need to change the default.

The radio acquisition (Fig. 2, red block) and the FFT algorithm (cyan block) run in parallel on two independent cycles (*threads*), where the first one runs at full speed while the second runs at the rate determined by the “Refresh interval” control. So, such rate affects the scrolling speed of the waterfall, non the acquisition speed that is always the maximum possible for the required number of



points.

When the interval chosen is too small, the FFT will re-process the same sample already processed in the previous loops. When this happens, the status line, placed at the bottom of the window shows the message “Acquisition time exceeded”. To avoid this, the resolution step must be enlarged and /or the logging level reduced. However, a more powerful hardware (multi-core) allows to reach higher rates.

The last control, “FFTW flag” allows to select the performance of FFT, the available values are “Estimate”, “Measure” and “Patient”.

Default is “Estimate” since it allows to start the FFT cycle quickly by setting the internal parameters of the algorithm on estimated values, without taking in count the performance of the host machine. With “Measure”, the algorithm calculates the FFT performance at run-time adjusting its parameters to best match the host machine, while the latter “Patient” takes even more time to achieve even better results, but it takes some more time at acquisition start, causing the GUI to freeze for some seconds after pressing the “Start” pushbutton before display the scans on the waterfall.

### 6.3.4 Output settings

The third tab “Output” includes the controls that have effect on file generation and peaks capture. There are three operating modes in which *Echoes* can operate, depending of how the following controls are set (see par.7). The content of the output data files reflects the operating mode chosen.

Let's consider “GNUplot output” first, because it affects the generation of the GNUplot's data dump files. When the operating mode is “automatic”, *Echoes* will catch waterfall screenshots in presence of S-N peaks. When “GNUplot output” is set, it dumps also on a **.DAT** file the scan data. Such file is particularly useful when running in console UI when no waterfall is displayed and screenshots are unavailable.

When the acquisition is stopped, a GNUplot's command file (**.PLT**) is generated. By opening that file with GNUplot, all the dumps caught in last acquisition session will be plotted sequentially.

More details about output files are explained later.

Now let's consider the first control, “Shot *xx* sec after peaks” that affects screenshots only, not GNUplot's data files. It delays the screenshot trigger by *xx* seconds after the peak event detection, to allow the event to be caught in its entirety on the waterfall.

“Stop after” specifies a number of screenshots that must be taken until the acquisition is stopped. With a zero value, only a GNUplot's data file is produced, containing all the scan caught until the acquisition's stop.

“Shot duration” affects only GNUplot's data dumps. While the time coverage of a screenshot is determined by the waterfall window size and the refresh interval chosen, the time coverage of a GNUplot's data dump must be selected thru this control. This is also the delta time between consecutive shots in periodic mode.

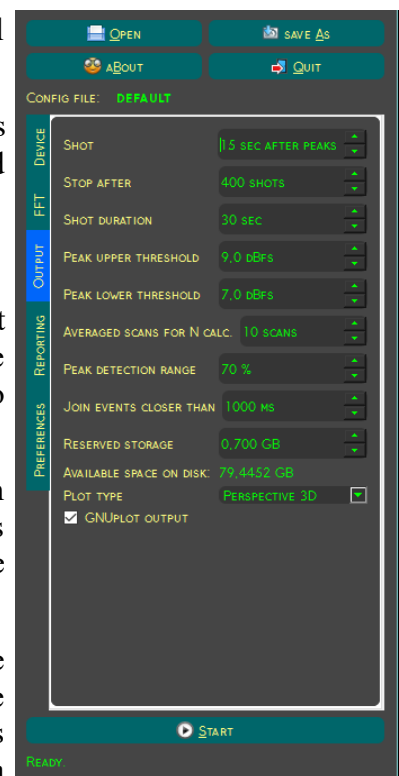


Fig. 8 Output tab



**Changed in 0.26:**

The “Peak Upper Threshold”, when set to non-zero value, activates the events detection. Thresholds can work in three different ways: “Absolute”, “Differential” and “Automatic”. The mode can be set in the “Preferences” tab (6.3.6).

The default mode is “Differential”, it matches the traditional way Echoes thresholds worked until now: the differential thresholds specify S-N values in dBfs. When the actual S-N value in the detection range exceeds the upper threshold, a new event is started, to stop when the actual S-N falls below the “Peak Lower Threshold”.

When the mode is set to “Absolute”, the thresholds are expressed as S values in dBfs. When the actual S value in the detection range exceeds the upper threshold, a new event is started, to stop when the actual S falls below the lower threshold. This mode could be the best choice for stations installed in quiet locations, without artificial disturbances and low and constant daily average noise (country, far from cities, airports, military installations etc.).

The “Automatic” thresholds are expressed as percentages. The programs calculates continuously a mobile average S-N value. A new event is triggered when the actual S-N exceeds the average S-N by the percentage specified as upper threshold, and the event ends when the actual S-N falls below the average S-N plus the lower threshold percentage. So in this mode the dBfs thresholds are recalculated at each scan, to cope with average S-N variations. The automatic mode has been thought for stations placed in noisy locations, where artificial radio disturbances are frequent and the average noise level varies significantly along the day. The automatic mode allows to reduce the number of fake detections, the price to pay is that weakest events could be lost.

The time elapsed between the crossing of the upper threshold and the falling under the lower threshold will determine the event lasting. These thresholds should be chosen carefully, because if the lower threshold is too low, the event started won't be never close.

“Noise filter” and “Peak detection range” controls work together. The detection range is a frequency interval, expressed in % of the band displayed in waterfall and centered on waterfall's central frequency and is represented by two blue arrows. *Echoes* calculates the noise N value by averaging, for each scan, all the FFT points falling in this range while the peak level S is the highest point value. The purpose of the detection range is to limit the detection of false positives due to planes, satellites, etc.

Unluckily, the narrower is this range and/or smaller is the point resolution, the wider is the variance of N, since the number of points considered for calculation of average becomes smaller. A wide variance of N causes the detection of fake events. In order to limit the variance, the control “Averaged scans for N calc” allows to specify the number of scans to be averaged to calculate a flatter N **and the average S-N reference value for automatic thresholds**. The bigger is the number, the flatter will be the cyan noise line on the power graph at left of waterfall.

The "Join time" control specifies the minimum time distance between consecutive echoes in order to be considered as a single event. This control is turning out really useful specially when working with scanning radars as signal source, like the *Graves* is.

“Reserved storage” allows to define an amount of disk space to be left free, in megabytes. When the free space falls below this limit, the acquisition stops spontaneously. Zero means no limits. The total

free space is continuously displayed just below.

Overdense events lasting more than 10s appear often as a dashed vertical path on the spectrogram. The previous releases of the program counted such paths as a succession of iperdense/ipodense events, making those counts less reliable.

The default value is 1000 ms. For GRAVES radar 4500 ms seems the best choice.

“Plot type” selects the format of GNUplot data files: “2D color-mapped” (like screenshots but more detailed), “3D perspective” and “2D total power”.

*Echoes* works exclusively with UTC time; all the time references present in the logs and in the waterfall are in UTC.

### 6.3.5 Report settings

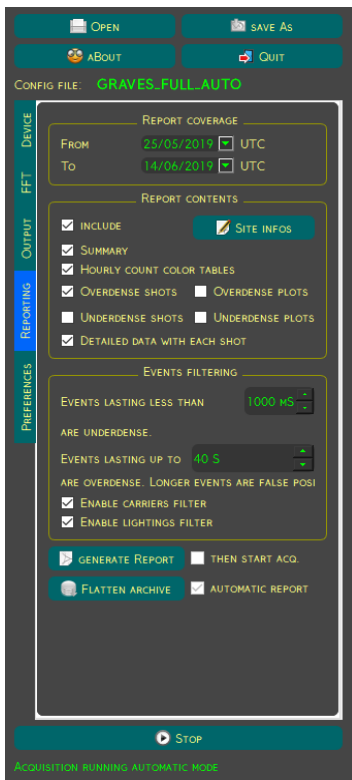


Fig. 9 The Report tab

There is a particular work that *Echoes* does each time the acquisition stops, later called “postprocessing”.

Its purpose is to move the daily data in dedicated archive subdirectories under working directory at midnight UTC, to subdivide the events and the relative shots and plots in overdense / underdense / fakes depending of the event lasting (see 7.5.) and finally produce the daily report table (7.4.6).

The *daily report* is a .csv table file containing the hourly count of the caught events, one day for row, splitted on 24 columns (hours) plus daily totals. This file is not closed at midnight. What is copied to archive is simply a snapshot, but the original file continues to grow in working directory day by day.

The events are collected by looking the daily scan files (7.4.4) (still .csv tables, recording statistic data about the events caught) present in working directory and referring the active configuration name.

Please note that reports can be generated only for automatic acquisition sessions. Other operating modes know nothing about “events”.

The “full report” can be created automatically, by checking “Automatic report” or upon user request, by pressing “Generate Report” in tab “Reporting”.

The full report is a long HTML file subdivided in 4 sections:

1. Site informations
2. Summary
3. Hourly count tables
4. Shots with statistic infos (overdense and/or underdense)

These sections can be a little personalized by acting on the following controls:

In the “Report coverage” box, must be inserted the period of days considered in the report. By default (brand new configuration cover file) the field “To” is set to the current date while the “From” date is one day before, in order to at least one day.

In “Report contents” box there are the following options:

“Include site infos”: by clicking the button, a dialog box appears, with some contents that can be customized: the station name, logo icon, position, setup and contact information, plus a short description of the report content. The data entered here are persistent and saved

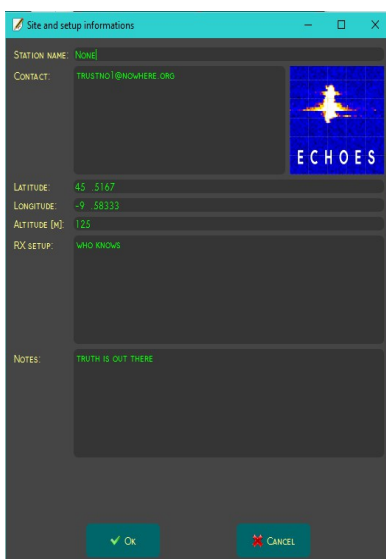


Fig. 10: Site infos dialog

in configuration file like most of other data entered in *Echoes*.

The logo selected here will appear also on the waterfall window (see Fig. 15 ) to personalize the screenshots taken. **New in 0.26:** The latitude and longitude are displayed at the bottom, in order to be included in screenshots.

The site info can be customized but if the voice is unchecked in “Report contents” these won't appear in the report.

The “Summary” info is a short description of the report content followed by a table where all the considered events – identified with the progressive number that appears in their file names – are subdivided in “Underdense”, “Overdense” and “Fakes” [*work in progress*]

“Hourly count color tables” these tables are structured like the “daily report” tables already seen, but they're also colored with a color scale to give a more intuitive impression of what happened in each hour of the considered days.

Totale senza falsi positivi:

ora UTC:	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	Totali
11/12/2017	19	12	10	21	17	21	20	22	19	20	21	12	9	12	14	9	6	9	23	27	31	32	40	38	464
12/12/2017	29	16	14	31	34	41	30	24	25	29	18	30	14	14	15	15	17	14	22	43	53	68	92	81	769
13/12/2017	50	23	19	36	61	42	58	47	34	29	25	27	36	33	12	11	17	21	43	54	18	0	0	0	696
14/12/2017	1	1	0	1	0	1	0	2	20	19	4	34	23	36	24	14	10	19	28	36	33	74	52	41	473
15/12/2017	33	34	29	31	35	28	33	25	24	35	25	33	49	26	25	27	25	31	15	38	46	22	25	29	722
16/12/2017	20	23	30	27	31	46	32	60	41	31	38	30	28	24	17	30	30	22	42	28	32	30	38	20	750
17/12/2017	22	31	13	25	19	43	27	41	40	25	48	63	38	27	41	47	23	20	27	25	42	25	15	15	742
18/12/2017	16	16	17	47	25	25	49	37	30	18	37	18	12	22	30	24	15	20	22	23	22	0	0	0	525

Fig. 11: Hourly count for Geminids 2017

The tables produced are three, the first is the total of overdense and undersense events without fakes. Follows a table of overdenses and a third of underdensities.

The report continues with the screenshots and/or the plots of all the events requested by checking “Overdense shots/plots” and “Underdense shots/plots”.

The last option “Detailed data with each shot” includes a table of statistic data along with each event reported.

The last box in “Reporting” tab “**Events Filtering**” is about the time thresholds that are applied to each event lasting, to discriminate underdense events from overdense ones and exclude false positives from statistic counts.

**New in 0.26:** two new experimental fakes filters have been added.

The “Carrier filter” is aimed to discriminate false positives due to radio carriers falling exactly at waterfall's center.

The “Lightings filter” detects false positives that alter the S-N along the entire waterfall's bandwidth, usually due to lightings or artificial electrostatic discharges.

Both these filters work by doing simple calculations on the values recorded on the statistic CSV: the lasting, the echo area, the full area and the number of peaks.

The checkbox “then start acq.” near the “Generate report” button allows the automatic restart of

acquisition once the report has been generated, since it can take a quite long time to complete.

The button “Flatten archive” allows to create a copy of the current configuration's archive hierarchy (Fig. 27) in a single folder to be selected through a dialog box, and removing all the subfolders keeping images and data files made in different days at the same level.

“Automatic report” : if checked, at midnight UTC the acquisition stops and – besides the daily report and archiving (7.4.6) a full report is also generated. That report covers the interval specified by “From” and “To” controls and these dates are automatically updated at the end of each report generation. Then the acquisition restarts afterwards.

Finally, the images linked in report are those stored in the archive folder. So, the *html* file should not be moved to other places alone without its archive folder.

### 6.3.6 Preferences

The last main window's tab is about user's preferences. Some people dislikes to see the measuring grids above the waterfall and the side graphs, so this tab allows to hide them by unchecking the related box.

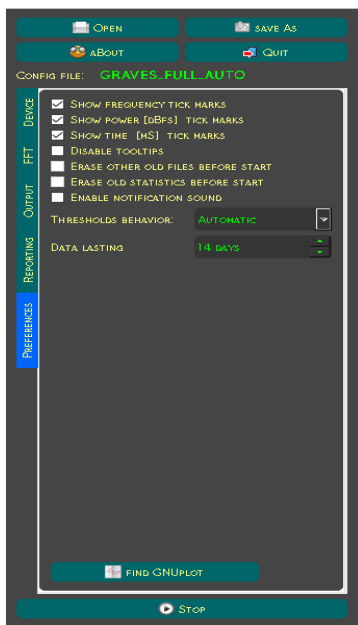


Fig. 12: Preferences tab

Even the tooltips can result annoying once the user has become familiar with *Echoes* controls, so they can be hidden by checking “Disable tooltips”.

“Erase old data before start” when set, all the data files present in the working directory produced with the configuration file actually loaded will be deleted at acquisition's start. This option is useful while experimenting with the parameters setup but remember to uncheck it before saving the final setup on a new configuration file!

“Erase old shots before start” : the previous options didn't erase the screenshots images but only the data files (.PLT .DAT .CSV). This option applies only on screenshots (.PNG) a. The considerations written above are still valid here.

“Data lasting” allows to set the maximum age of output data files, to preserve disk space. Older files will be automatically deleted at acquisition's start. This applies only on the files generated with the configuration file actually loaded (its name is showed just below the top buttons) leaving others untouched.

The button “Find GnuPlot” appears when *Echoes* cannot find GnuPlot by itself. When pressing the button, an “Open File” dialog is opened to select the correct GnuPlot's executable file (**wgnuplot.exe** under Windows, or simply **gnuplot** under Linux).

“Enable notification sound”: checked by default, it can be unchecked to avoid playing the notification sound (the “ping”) each time a new event is started.

**New in 0.26:** the “Thresholds behavior” selector allows to choose the preferred mode for thresholds: Absolute, Differential (default) and Automatic, see 6.3.4.

### 6.3.7 Status bar

It's a short variable text at the bottom of the window describing the program status. Due to window's shape, long text gets truncated, but they can be read entirely in the *tooltip*, a little box with suggestions that appears by hovering a control with the mouse. Note that tooltips must be enabled in “Preferences” tab first (see 6.3.6).

Normally the displayed text is “Ready” meaning that the program is ready to accept new commands. When acquisition is running, this bar shows the operating mode active (continuous / periodic / automatic) (7.1.).

Another message that can appear is “Acquisition time exceeded”, meaning that the “Refresh interval” parameter chosen (6.3.3) is too small and refresh cannot occur so fast.

## 6.4. The Waterfall window

### 6.4.1 Graphic panes

This window, that represents the output of FFT processing, split in 3 panes:

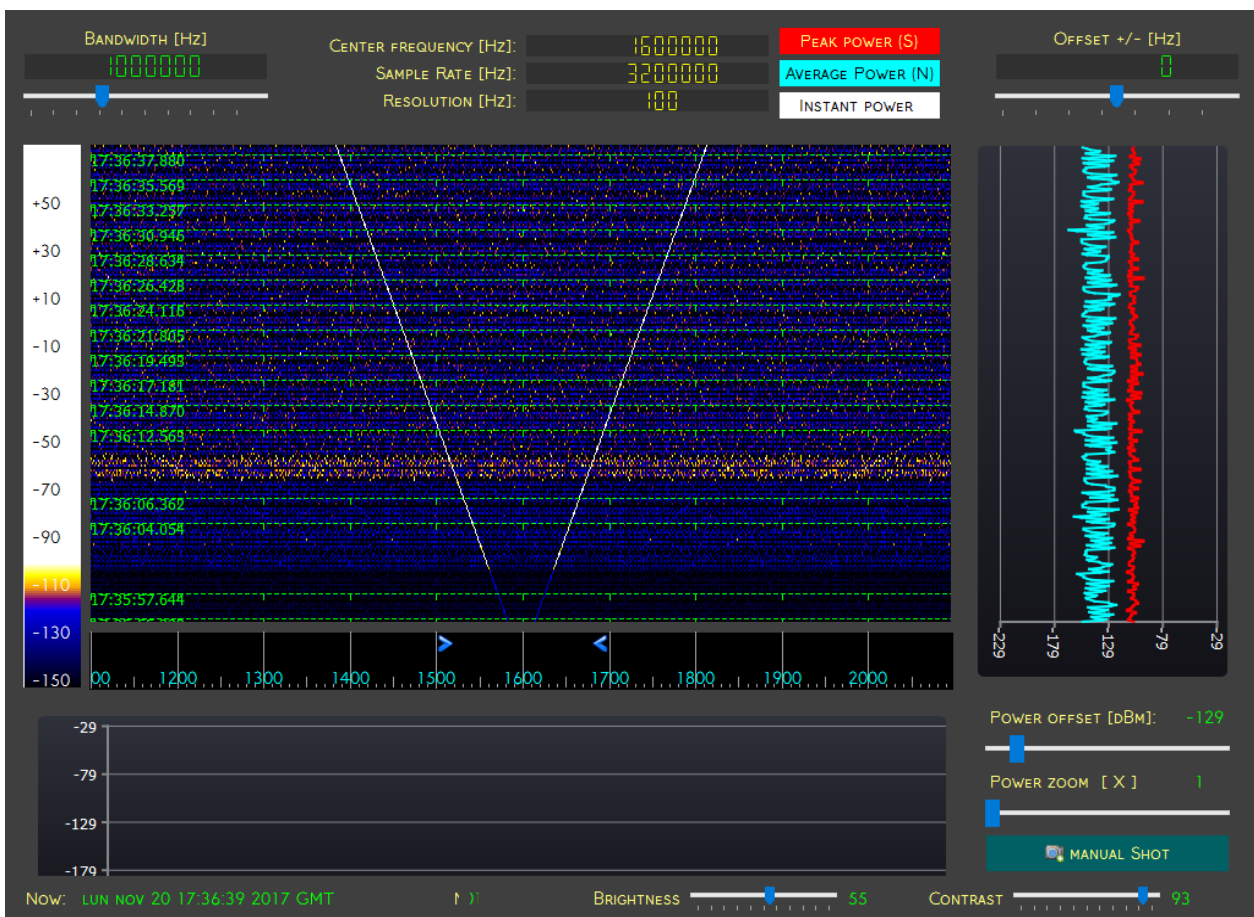


Fig. 13: Waterfall window while running the test pattern in continuous mode

Figure 1:

At the center is the “Waterfall spectrogram”, with the time axis in Y and frequency in X, scrolling downwards. At its left is the color scale, it can be adjusted with the “Brightness” and/ “Contrast” sliders placed at window's bottom.

Depending of the sampling rate, the bandwidth chosen and the window's size, each pixel can represent one FFT point or the maximum value of  $n$  adjacent FFT points because narrow resolutions (for instance 1.91 [Hz@250kHz](#) SR) produces many thousands of points ( $2^{17} = 131072$ ) while the screen resolutions generally varies around few thousands of pixels in width. For this reason, a change of resolution or sampling rate produces a change of noise level (N) that affects the background color, that needs to be readjusted acting on the sliders.

While scrolling, horizontal time tick marks are plotted in green (color not present in color scale) with UTC time printed at the left side. The waterfall is in fact a flat representation of 3D data, since the dBfs values are represented by colors instead of points on a Z axis.

At the right side, a scrolling graph called “Power history” represents the average power of each scan line (in cyan) aka “noise value” (abbreviated “N”) and the maximum value (in red) aka “peak power” (abbreviated “S”) . The graph scrolls at the same rate and direction of waterfall.

The S & N values are not calculated on each entire scan, but on a portion of it, centered at waterfall's center frequency (displayed above) and adjustable symmetrically in Output tab ( “Peak selection range”) as percentage. This range is represented by the blue arrows just below the waterfall, on the frequency ruler and can be enlarged to cover up to the entire scan (100%). The N value is the result of an averaging process, that can be adjusted in main window's Output tab (6.3.4).

Finally, below the waterfall is the “Instantaneous power” graph, where the FFT points values are represented in Y by frequency in X and is not affected by the selection range.



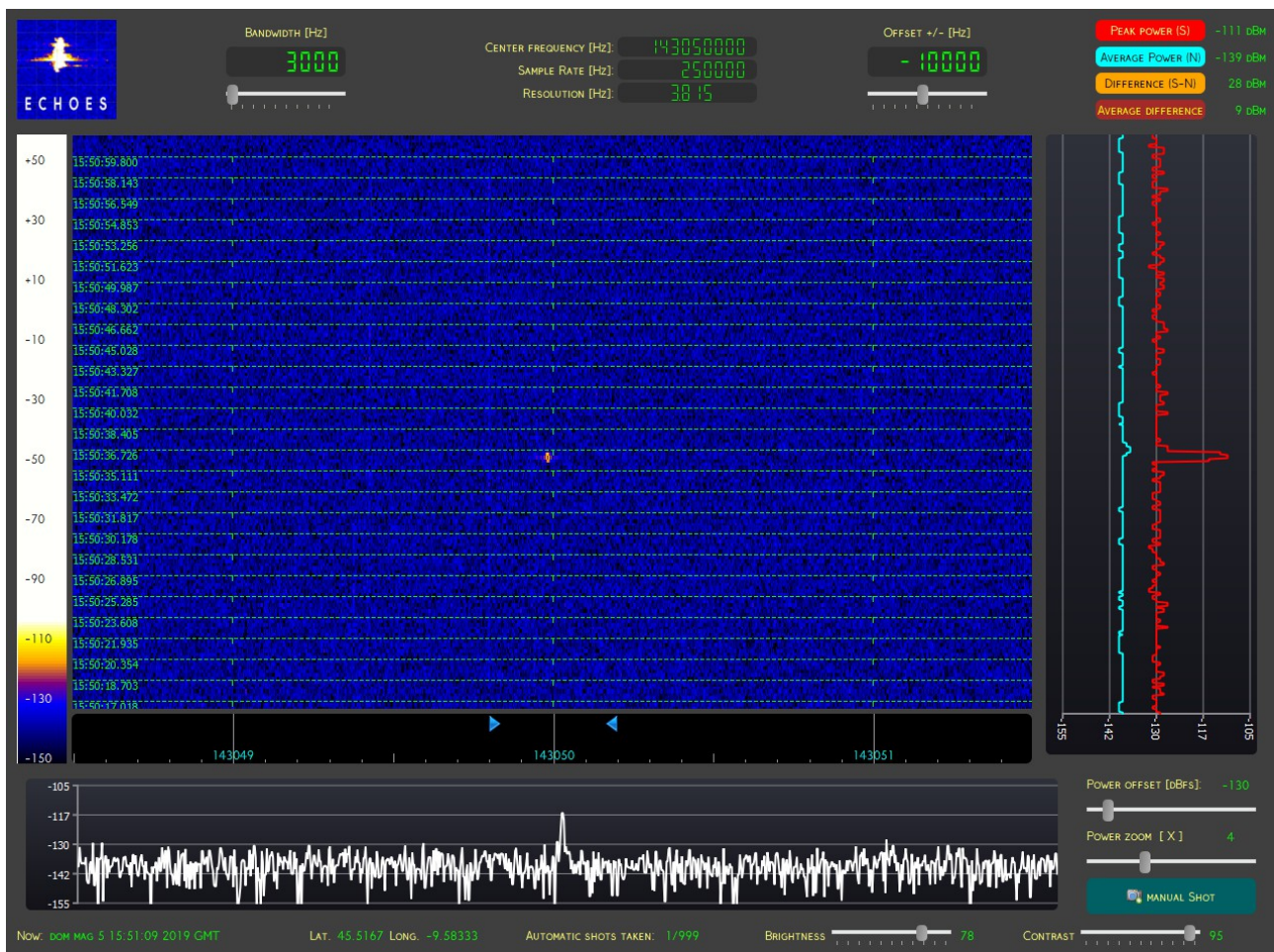


Fig. 14: Waterfall screenshot of a real meteor echo of GRAVES radar

### 6.4.2 Waterfall controls

The controls at top of the window affect the waterfall and the “Instant power” graph. The frequency range showed in the waterfall can be arranged thru the two sliders “Bandwidth” and “Offset” placed at the left and at the right of the window. By default, each time the “Tune” control is changed in main window (see 6.3.2) the bandwidth displayed is set equal to sample rate, while offset is set and blocked to zero.

Now, by acting on “bandwidth” slider, the frequency range displayed shrinks; the frequency ruler at bottom reflects immediately these changes. The “offset” slider gets unlocked now and it can slide up/down this range, so its center frequency will shift respect to the tuned frequency. This waterfall's center frequency is displayed at the top, with the actual sampling rate and resolution too, in order to catch these setting in screenshots.



### 6.4.3 Graphs controls

The “Power zoom” and “Power offset” sliders, at right/bottom corner of the window, have effect on “Power history” and “Instant power” graphs. The zoom slider reduces/enlarges the dBfs range represented, while the offset slider scrolls the represented range in both directions.

Immediately below the power sliders, there is the “Manual shot” pushbutton. It triggers a waterfall snapshot when pressed. Manual shots are counted separately from automatic snapshots, they do not produce GNUplot data output like automatic does and the even the manual snapshot files are named with a different prefix.

### 6.4.4 Notch filters

By right-clicking on the frequency ruler, it's possible to place a notch filter in that frequency. A small pop-up window appears, to set the width of the blocked band in Hz. The value can be roughly set by the means of a slider or edited with the keyboard. Once confirmed the value, a red triangle

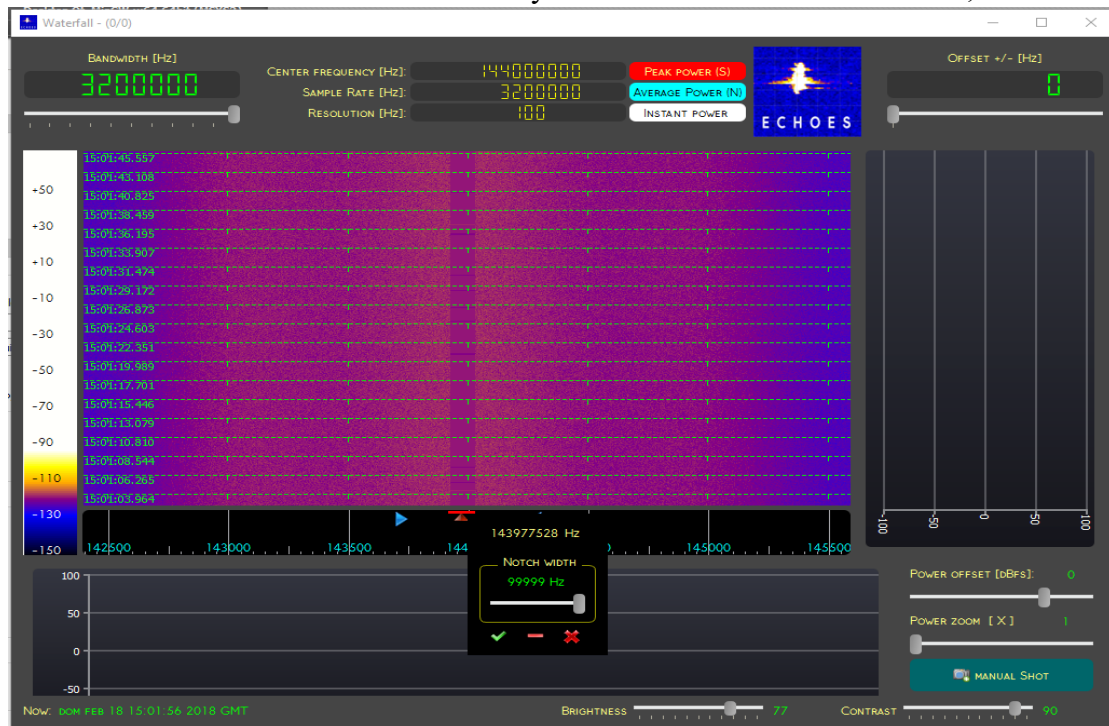


Fig. 15: Setting / editing a notch filter.

appears, with a horizontal red line above it, to visualize the blocked band width.

It's possible to add many filters, the unique condition is that they cannot overlap each other. The effect of notch filter is that, in the frequency range it covers, the waterfall will display the N value calculated in the last scan, in place of the real value produced by FFT. This effect applies also to all the data and plot files produced.

The filter can be edited by simply right-clicking on the red arrow, and deleted by clicking the minus icon on the pop-up.

## 7 Playing with *Echoes*

### 7.1. Operating modes

There are three operating modes in which *Echoes* can operate, but there is not a single specific GUI control to perform this switch; the operating mode in fact is selected by a combination of settings in main window:

Stop after X shots	Peak upper threshold	Resulting operating mode
Zero	Zero	Continuous
Non-zero	Zero	Periodic
Non-zero	Non-zero	Automatic

Tab. 1: Operating modes truth table

In case the settings operated in main window do not allow to select an operating mode or determine an incongruent configuration, the message “Wrong parameter” appears in the status bar, at bottom of main window.

#### 7.1.1 Continuous recording

When “Stop after...” and “Peak upper threshold” are both set to zero, the program records continuously the statistics of each scan in the statistic CSV file (7.4.4) and, if GNUplot has been enabled, even the plot file does the same. This mode is suited to record the total power (plot type “power 2d”) in an unique continuous plot as showed in Fig. 25. Other types of plot should not be tried in this mode since they will tend to grow up to unmanageable sizes.

#### 7.1.2 Periodic recording

By setting a value in “Stop after...”, *Echoes* will trigger shots at regular intervals (“Shot duration”), but is still not able to detect events. In statistic CSV file is saved one line for each scan, as in continuous mode, but the name of the related screenshot is saved along with each scan line. Plotting – if enabled – is no more continuous but subdivided in numbered shots.

#### 7.1.3 Automatic snapshot recording

When “Stop after...” and “Peak upper threshold” are both set to nonzero values, *Echoes* will detect peak events: when the difference (S-N) crosses the upper threshold, a countdown (“Shot xx seconds after peak”) starts. When it expires, it triggers a screenshot and a plays a “ping” notification sound. To be able to catch long events, this delay should be set in order to trigger just few seconds before the peak slides out of the screen.

While taking the screenshot, the “Instantaneous power graph” replays the power data recorded when the peak was detected, so the shot will simultaneously record the power peak in frequency domain, in time domain (“Power history” graph) and both (waterfall).

Simultaneously, when GNUplot output is enabled, the word “CAPTURING” appears below the waterfall and *Echoes* starts storing scan data in memory until (S-N) falls under the lower threshold,

closing the event. The word CAPTURING disappears and - in this moment - the plot data in memory are stored in a GNUplot data file, numbered as the relative screenshot.

The statistic CSV file content is different in automatic mode than in the other two. Instead of store a line of statistic data at each scan displayed in waterfall, the writing is executed only at raising front, at event peak and falling front, so only 3 lines are written on disk for each event. The third line includes also the results of some calculations executed when the event is active, like the total lasting and the “area” in pixels covered by the echo on the waterfall.

Such “area” will be used in future by the postprocessing algorythm to discriminate the events. Actually the discrimination is based only on event lasting.

## 7.2. From FFT output points to pixels

The number of frequency points produced by the FFT algorithm is expressed by the ratio between the sample rate and the resolution chosen. The number of these points can be much bigger than the waterfall's width in pixels, therefore the scan data are averaged and scaled to this size (the thin purple arrow in Fig. 16) still remaining centered on the tune frequency (the pink arrow).

Nevertheless, the waterfall could display only a portion of the entire scan; in fact for meteor scatter purposes this selection is normal practice, since the maximum spectral extension of a meteor echo is less than 10kHz and working on a part of the spectra makes the echo more evident in screenshots and plots. This selection is called “bandwidth” and is represented by the big blue arrow.

This selection can be shifted up/down on the full scan by an “offset” (brown arrow) relative to the tune frequency. The sum of tune frequency and this offset gives the “center frequency” displayed above the waterfall.

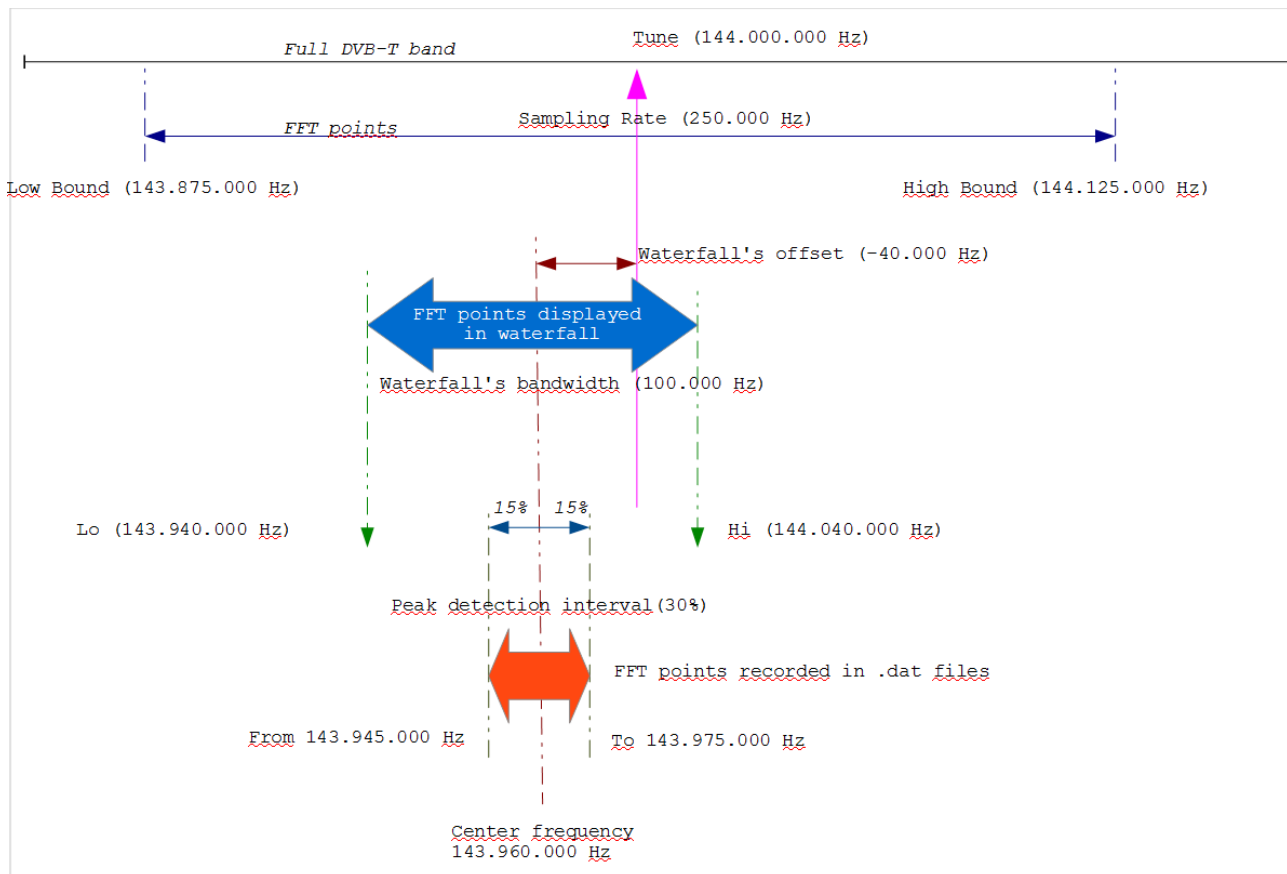


Fig. 16: Sample portion of spectra displayed in waterfall and recorded on plot files

The “peak detection range” (par.6.3.4) selects a narrower portion of the bandwidth (orange arrow). The data recorded on plot files are about all the pixels falling in this range.

### 7.3. Examples

The following images (in italian) have been taken with Echoes v.0.17 between Dec11 and Dec18, 2017 under Geminids shower.

My station is built around a RTL-SDR receiver that's somewhat more sophisticated than usual DVB-T dongles: the *Nongles N3* [rif.14] .

The antenna is a 3-elements Yagi and in the middle there is a 88-108Mhz notch filter to limit the interference caused by commercial broadcasting radios.

The tune frequency was 143.060 kHz and the offset was -9900 in order to get the GRAVES radar carrier frequency almost in the middle of my peak detection range (12%) (“almost” because I didn't set the ppm error correction for my device yet).

The sample rate was 250kHz, the minimum frequency my device allows. Other important parameters are the rx gain set to maximum (49dB), the FFT resolution (1,9 Hz), the refresh interval (80mS), shot duration (30s), shot 15s after peak detection and the thresholds: upper 13 dBfs and lower 7 dBfs. Finally the bandwidth shown in waterfall was 5kHz.

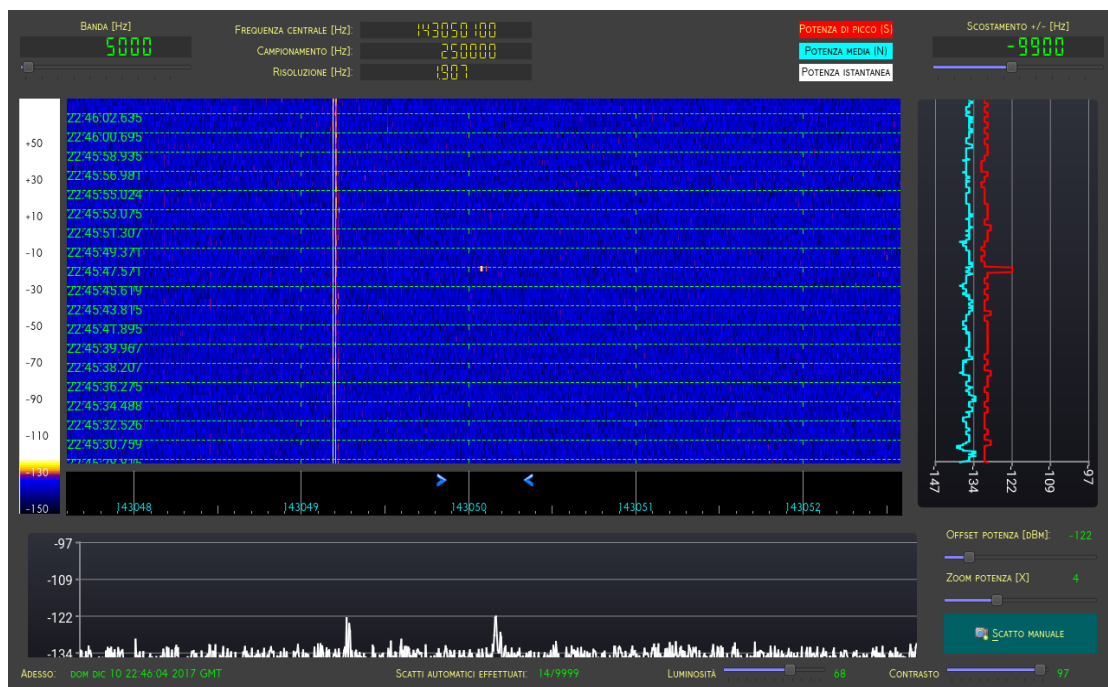


Fig. 17: Underdense event

(Now, my advice is to take a look of the IMO web page [ref. 6 at page 5] then continue reading from here)

Most of captured events look like the small dot in Fig. 17.

The straight vertical line at left is an undesired source, it can be external – a real radio signal coming from the air – or an interference produced by some electric and electronic device near the station: monitors, PCs, TVs, motors, xmas lights... Such noises can be easily hidden by setting a

digital notch filter (6.4.4) since they have fixed frequencies.

The straight line could also be a real radar reflection caused by an airplane. These echoes look as straight continuous lines that can move slowly toward left or right of the spectra to disappear suddenly after some minutes.

The purposes of a narrow peak detection range are two: the first is to limit unwanted event triggering caused by such echoes, even if they aren't completely maskable. The second is to limit the amount of data recorded in *.dat* files for GNUplot, since the pixels not included in the interval won't be stored.

Despite that unwanted signal, the image shows also a real meteor echo: that little white dot in the center of the spectrogram that triggered the event.

The so-called underdense events, like this, are produced by small meteors that burn quickly producing very little plasma and remaining invisible to human eyes (magnitude > 6).

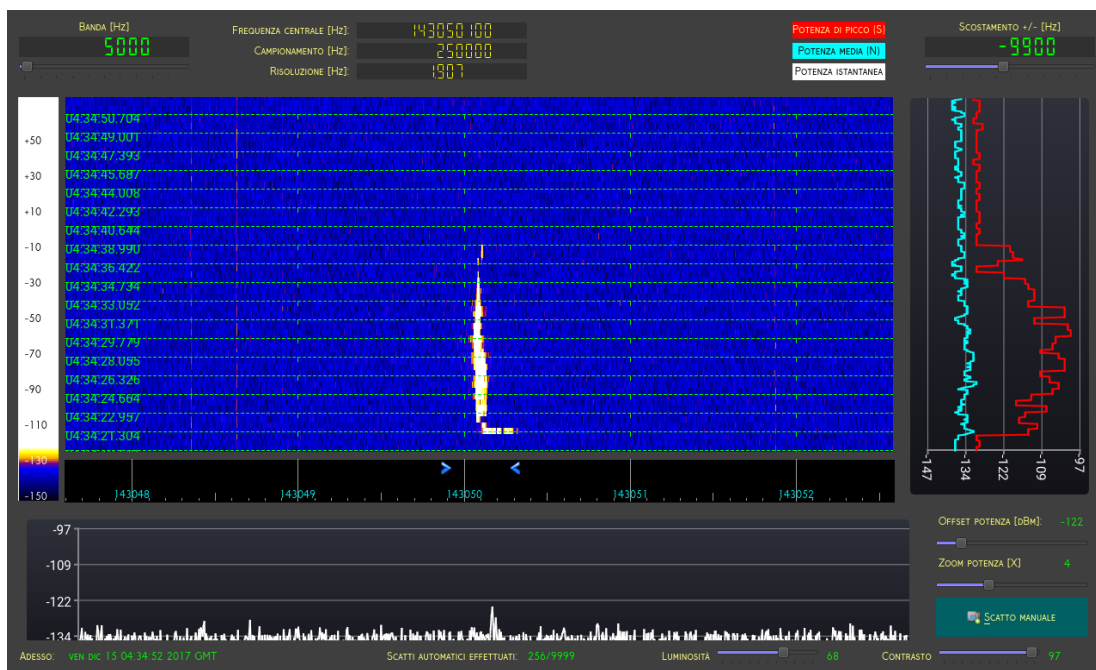


Fig. 18: *Overdense and long lasting event*

The screenshot in Fig. 18 shows an overdense event. Such events are generally attributed to more massive bodies or fireballs.

The horizontal pattern is index of the body's speed while getting far from station (Doppler effect with frequency decreasing) while its vertical pattern tells us the persistency of ionization caused by body's burning, lasting several seconds in the screenshot.

Sometimes, horizontal and vertical patterns are joined in curious ways, like in Fig. 19 that looks like an overlapping of two separated events with different patterns.

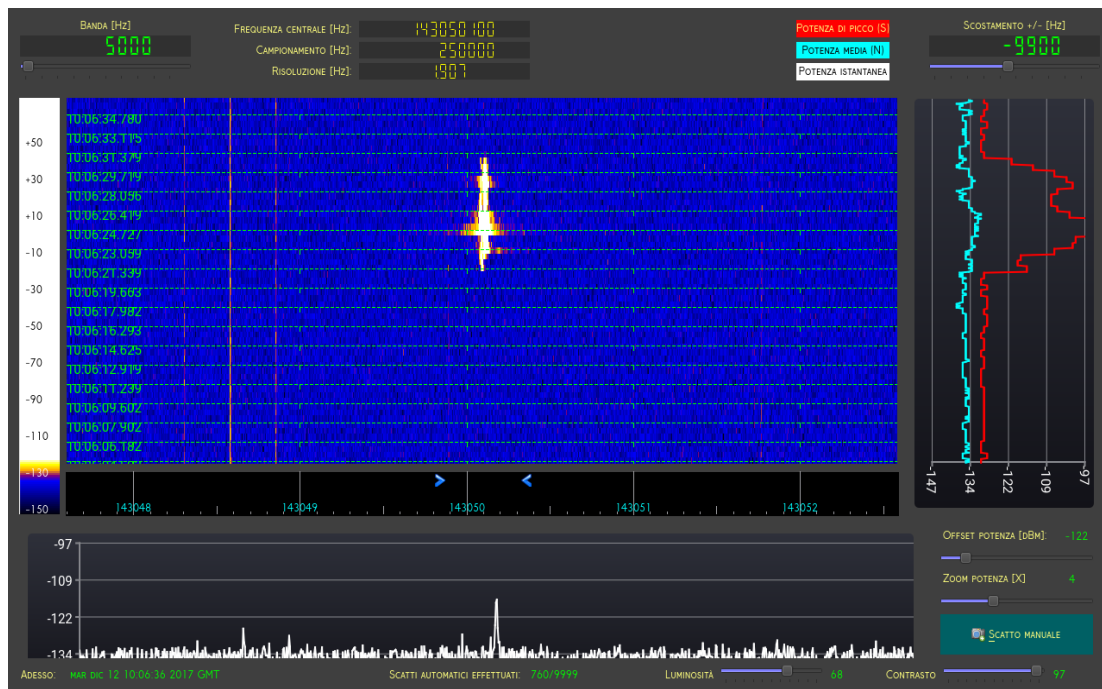


Fig. 19: Maybe the result of two overlapping events

When producing 2D color-mapped plots, the same event appears as in Fig. 20 when opened with GNUplot. The frequencies on X axis cover the peak detection range, marked by the blue arrows, while on Y axis there are the seconds elapsed since acquisition start (the date/time included in filename and showed above the plot). More info about GNUplot files are at par.7.4.5

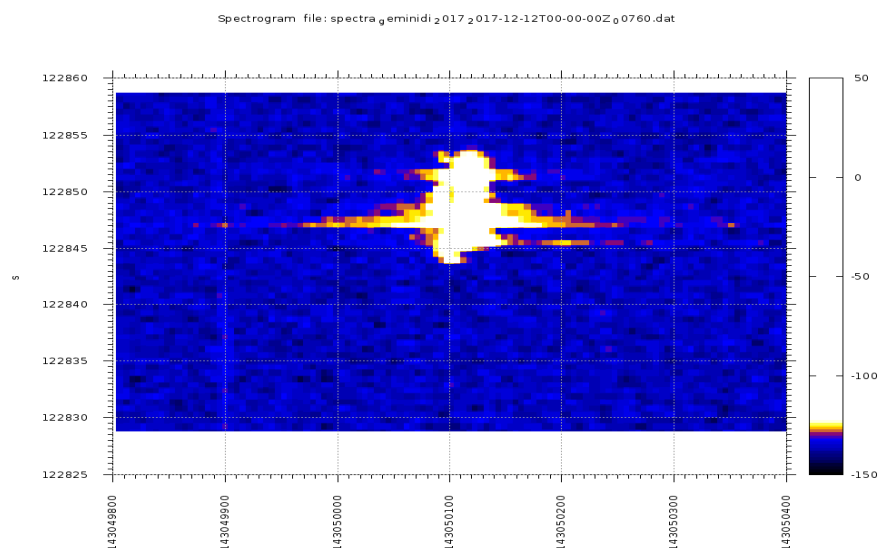


Fig. 20: The same event how it looks plotted by GNUplot in 2D color map



In some overdense events the ionized “burning tail” could lack completely, like in Fig. 21 . It seems that the meteor hit the GRAVES signal before burning. This could happen because the GRAVES is a radar that scans the sky in a known sequence so it could be that the signal illuminated the meteor while approaching the Earth but when it burned, the signal was sent in another sky sector. Or it could also be that the burning has been reflected somewhere far from my antenna so I could not detect it.

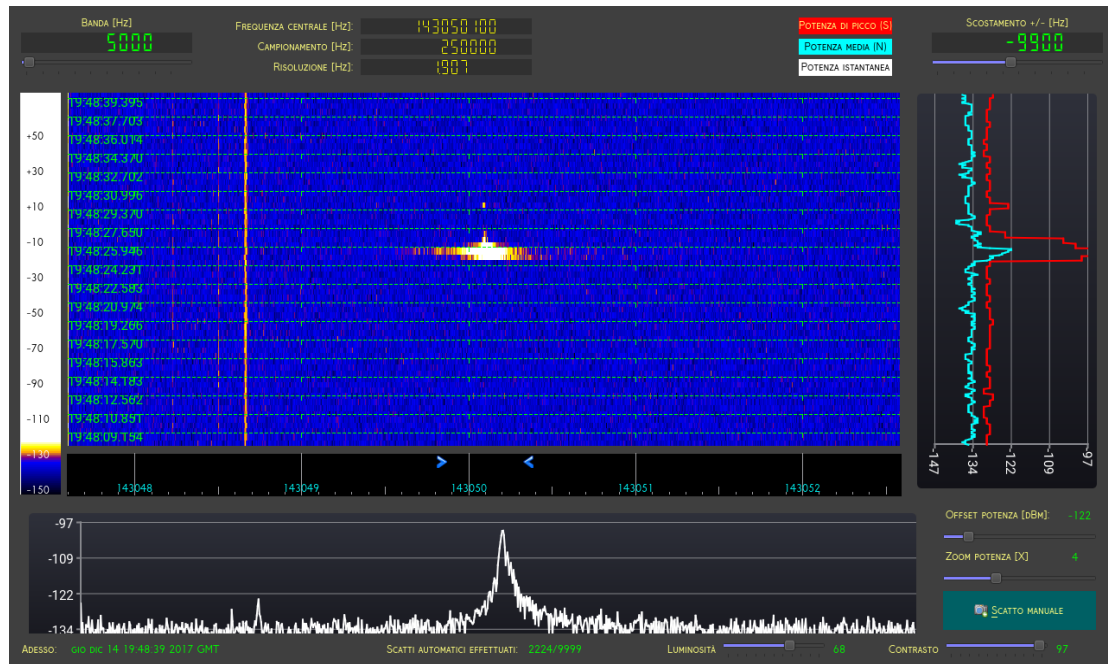


Fig. 21: Short lasting overdense event



The last sample (Fig. 22) shows two frequent source of fake events: the International Space Station (oblique dashed line) and a plane (oblique continuous line).

It's quite easy to check if the ISS was transiting above our station when receiving such dashed line by looking the ISS ephemerides on websites like “Heavens Above”

(<http://www.heavens-above.com/>) , while for planes it's not so easy to associate planes to fake events, due to the great number of planes flying in the sky sectors illuminated by Graves (check on <https://www.flightradar24.com> for instance).

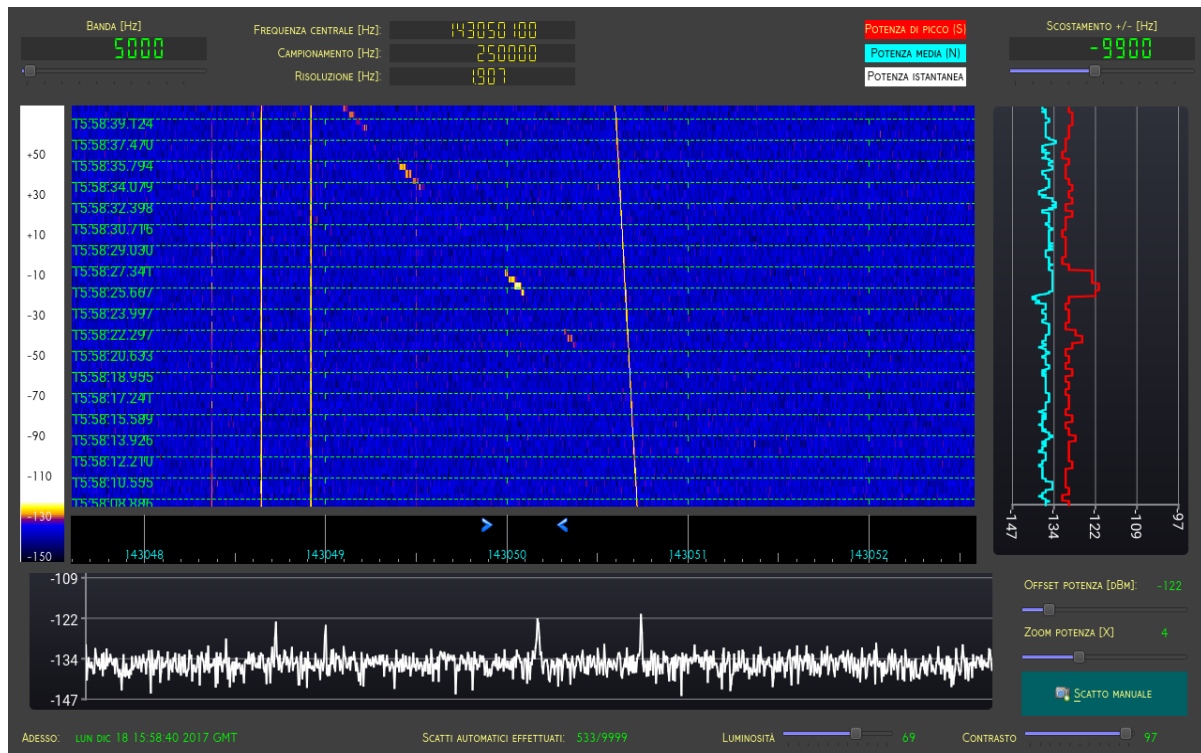


Fig. 22: ISS and plane

## 7.4. Files contents and naming conventions

*Echoes* reads and writes the following files while it runs:

### 7.4.1 Configuration files

The configuration files are text files that can be created by pressing the “SaveAs” button and giving the new file a new name. These file contain setup data in format **key=value** one per row. If needed, they can be patched by hand with a text editor. In case of erroneous data entered (for instance, a value out of allowed range) *Echoes* will replace it with a default hardcoded setting.

### 7.4.2 Log file

The file *echoes.log* is a system log created in the working directory, overwriting a previous one if present. It could be interesting only for debugging purposes, to track what the program were doing just before a crash. With **-n** parameter at command line, its verbosity can be incremented when more details are needed. The defined levels are:

-n	mnemonic	meaning
0	NONE	The log file is never created
1	FATAL	Logs only fatal errors that cause immediate crash
2	CRITICAL	Logs also critical errors causing loss of functionalities
3	WARNING	Logs also warnings about anomalous situations that could cause future loss of functionalities
4	INFO	Logs also some details about the internal state of the program that are not presented on GUI
5	DEBUG	Logs also cryptic and verbose messages for development/debugging purposes

Tab. 2: logging levels

On Linux platform, *Echoes* uses the system logging daemon facility **syslog** for logging purposes, so *echoes.log* won't be created to avoid redundancies.

The text sent to the logger can be sent to the standard output also by specifying the parameter **-v** in the command line.

### 7.4.3 Screenshots

They are .png photos of the waterfall window produced automatically while acquisition runs in automatic or periodic mode. Their file name are coded as follows:

*autoshot\_<configuration name>\_<acquisition mode>\_<date in ISO-8601 format>\_<nnnnn>.png*

where the last 5 digits are the event/screenshot reference number (column 1 in statistic CSV).

The screenshots can be required manually in any moment, regardless the acquisition mode selected, by pressing the button “Manual shot” at bottom/right corner of the waterfall.

Manual shots numbers follow a different progression (*mmmmm*) from automatic ones (*nnnnn*) and are named as follows:

*manshot\_<configuration name>\_<acquisition mode>\_<date in ISO-8601 format>\_<mmmmm>.png*

#### 7.4.4 Statistic CSV table file

AKA “scan CSV”, these files are labeled as:

*scan\_<configuration name>\_<acquisition mode>\_<date in ISO-8601 format>.csv*

they are textual semicolon-separated values tables and each line contains 27 cells/columns described below. The letter refers to the column ID when the file is opened in a spreadsheet :

- (A) Event number: while working in continuous or periodic mode this is a progressive scan number and each row gets a different number (A scan can be seen as a complete line of data in waterfall, while watching it in full bandwidth). Numbering starts from 1 at each acquisition session.  
  
Under automatic mode instead, the program gets the capability of events detection so the columns stores the progressive event number. Since always 3 lines are produced for each event (the raising front, the peak and the falling front) these lines will be marked with the same number.
- (B) Date of scan/event in local short format.
- (C) UTC time including milliseconds.
- (D) Lowest Hz: lowest frequency covered by waterfall at full bandwidth.
- (E) Highest Hz: highest frequency covered by waterfall at full bandwidth.
- (F) Bandwidth: waterfall's bandwidth, difference between columns (E) and (D).
- (G) Step Hz: FFT resolution or waterfall's granularity.
- (H) Thresholds mode : the next columns (I) and (J) are filled only if “Automatic” mode has been selected.
- (I) Auto up threshold % : upper threshold percentage in automatic mode.
- (J) Auto dn threshold % : lower threshold percentage in automatic mode.
- (K) Up threshold [dBfs]: upper threshold in dBfs. When the *S* value (O) exceeds this value and the thresholds mode (H) is “Absolute”, a new event starts. For other thresholds modes instead, this value is compared against the difference *S-N* (Q) .
- (L) Dn threshold [dBfs]: lower threshold in dBfs. When the *S* value (O) falls below this value and the thresholds mode (H) is “Absolute”, the active event ends. For other thresholds modes instead, this value is compared against the difference *S-N* (Q).
- (M) Range low Hz: lowest frequency of event detection range.
- (N) Range high Hz: highest frequency of event detection range.
- (O) Top peak *S* [dBfs]: highest signal got in the detection range.

- (P) Average  $N$  [dBfs]: filtered average value in the detection range.
- (Q) Difference  $S-N$  [dBfs]: instantaneous difference between columns (O) and (P).
- (R) Average difference  $S-N$  [dBfs]: mobile average of the last  $n$  (Q) values. The value  $n$  is the same used to calculate the  $N$  value (P).
- (S) Peak Hz: frequency where the peak signal (O) has been found.
- The following columns are always generated in order to keep the number of columns fixed, but they're meaningful only in automatic event detection mode, on falling front:
- (T) Lasting [mS]: time difference between the falling and raising fronts of the event. This value is nonzero only in falling front rows, for obvious reasons.
- (U) Freq shift [Hz]: difference between the values (S) of falling and raising fronts, to calculate Doppler's effect.
- (V) Echo area [pixels]: is an experimental field aimed to improve fake events discrimination. This value is nonzero only in falling front rows. The number of pixel adjacent to the (S) with value higher than the lower threshold (L) are summed and this operation is repeated until the falling front is found. The resulting sum should give, in numerical form, the “extension” of the event, besides its intensity that is already represented by (Q).
- (W) Interval area [pixels] : is an experimental field aimed to improve fake events discrimination. This value is nonzero only in falling front rows. It's the area in pixel of the rectangle having as height the event lasting and as width the events detection range.
- (X) Peaks count: number of points, in the entire event, that cross the upper threshold, besides the maximum one.
- (Y) LOS speed [m/s]: apparent speed (*Line Of Sight*) calculated on the doppler shift (U), taking as reference the central frequency (tuning+offset)
- (Z) Event status: raise, peak, fall.
- (AA) Shot name: Screenshot file name related to this event. They are progressively numbered and the number must match what in column (A).

In order to avoid problems while importing in spreadsheets, the decimal point could be a point or a comma, depending of the system localization. The following few lines have been extracted from a statistic CSV file running in continuous mode:

```
Event nr.;Date;UTC Time;Lowest Hz;Highest Hz;Bandwidth;step Hz;Threshold mode;Auto up threshold %;Auto dn threshold %;Up threshold
[dBfs];Dn threshold [dBfs];Range low Hz;Range high Hz;Top Peak (S) [dBfs];Average noise (N) [dBfs];Difference (S-N) [dBfs];Average
difference (S-N) [dBfs];Top Peak Hz;Lasting [mS];Freq shift [Hz];Echo area [pixel];Interval area [pixel];Peaks count;LOS speed
[m/s];Event status;Shot name
1;02/05/2019;00:03:51.991;143048478;143051478;3000;3,8147;automatic;75;70;18,245;17,7238;143049790;143050210;-120,141;-
139,719;19,5781;10,4257;143.049.939;0;0;0;0;0;0;-;
2;02/05/2019;00:08:31.949;143048478;143051478;3000;3,8147;automatic;75;70;16,8261;16,3453;143049790;143050210;-122,902;-
140,079;17,1774;9,61491;143.049.958;0;0;0;0;0;0;-;
```

The columns are always 27 (there are still 26 column separators followed by carriage return at the end) but the last column (shot name) is empty, since no shots are taken in continuous mode. So, the columns from (T) to (AA) don't contain meaningful values.

```

Event:dn,Date,UTC time;Lowest Hz;Highest Hz;Bandwidth;step Hz;Threshold mode;Auto up threshold %;Auto dn threshold %;Up threshold [dBfs];Dn threshold [dBfs];Range low Hz;Range high Hz;Top Peak [S] [dBfs];Average noise [N] [dBfs];Difference (S-N) [dBfs];Average difference (S-N) [dBfs];Top Peak Hz;Lasting [mS];Freq shift [Hz];Echo area [pixel];Interval area [pixel];Peaks count;LOS speed [m/s];Event status;Shot name
1:04/05/2019;00:03:51.991;143048478;143051478;3000;3,8147;automatic;75;70;18,245;17,7238;143049790;143050210;-120,141;-139,719;19,5781;10,4257;143.049.939;0;0;0;0;0;-;autoshot_GRAVES_periodic_2019-05-03_00001.png
2:04/05/2019;00:08:31.949;143048478;143051478;3000;3,8147;automatic;75;70;16,8261;16,3453;143049790;143050210;-122,902;-140,079;17,1774;9,61491;143.049.958;0;0;0;0;0;-; autoshot_GRAVES periodic_2019-05-03_00002.png

```

```

Event nr.;Date;UTC Time;Lowest Hz;Highest Hz;Bandwidth;step Hz;Threshold mode;Auto up threshold %;Auto dn threshold %;Up threshold [dBfs];Dn threshold [dBfs];Range low Hz;Range high Hz;Top Peak [S] [dBfs];Average noise [N] [dBfs];Difference (S-N) [dBfs];Average difference (S-N) [dBfs];Top Peak Hz;Lasting [mS];Freq shift [Hz];Echo area [pixel];Interval area [pixel];Peaks count;LOS speed [m/s];Event status;Shot name

1;04/05/2019;00:03:51.991;143048478;143051478;3000;3,8147;automatic;75;70;18,245;17,7238;143049790;143050210;-120,141;-139,719;19,5781;10,4257;143.049.939;0;0;0;0;0;0;0;Raise;autoshot_GRAVES_FULL_AUTO_automatic_2019-05-04_00001.png

1;04/05/2019;00:03:51.991;143048478;143051478;3000;3,8147;automatic;75;70;18,245;17,7238;143049790;143050210;-120,141;-139,719;19,5781;10,4257;143.049.939;0;0;0;0;0;0;0;Peak;autoshot_GRAVES_FULL_AUTO_automatic_2019-05-04_00001.png

1;04/05/2019;00:03:52.111;143048478;143051478;3000;3,8147;automatic;75;70;21,3741;20,7635;143049790;143050210;-120,141;-139,595;19,4541;12,2138;143.049.939;155;0;3;405;1;0;Fall;autoshot_GRAVES_FULL_AUTO_automatic_2019-05-04_00001.png

1;04/05/2019;00:08:31.949;143048478;143051478;3000;3,8147;automatic;75;70;16,8261;16,3453;143049790;143050210;-122,902;-140,079;17,1774;9,61491;143.049.958;0;0;0;0;0;0;0;Raise;autoshot_GRAVES_FULL_AUTO_automatic_2019-05-04_00002.png

2;04/05/2019;00:08:31.949;143048478;143051478;3000;3,8147;automatic;75;70;16,8261;16,3453;143049790;143050210;-122,902;-140,079;17,1774;9,61491;143.049.958;0;0;0;0;0;0;0;Peak;autoshot_GRAVES_FULL_AUTO_automatic_2019-05-04_00002.png

2;04/05/2019;00:08:32.048;143048478;143051478;3000;3,8147;automatic;75;70;17,9932;17,4791;143049790;143050210;-122,902;-139,979;17,0773;10,2818;143.049.958;75;0;2;270;1;0;Fall;autoshot_GRAVES_FULL_AUTO_automatic_2019-05-04_00002.png

```

37

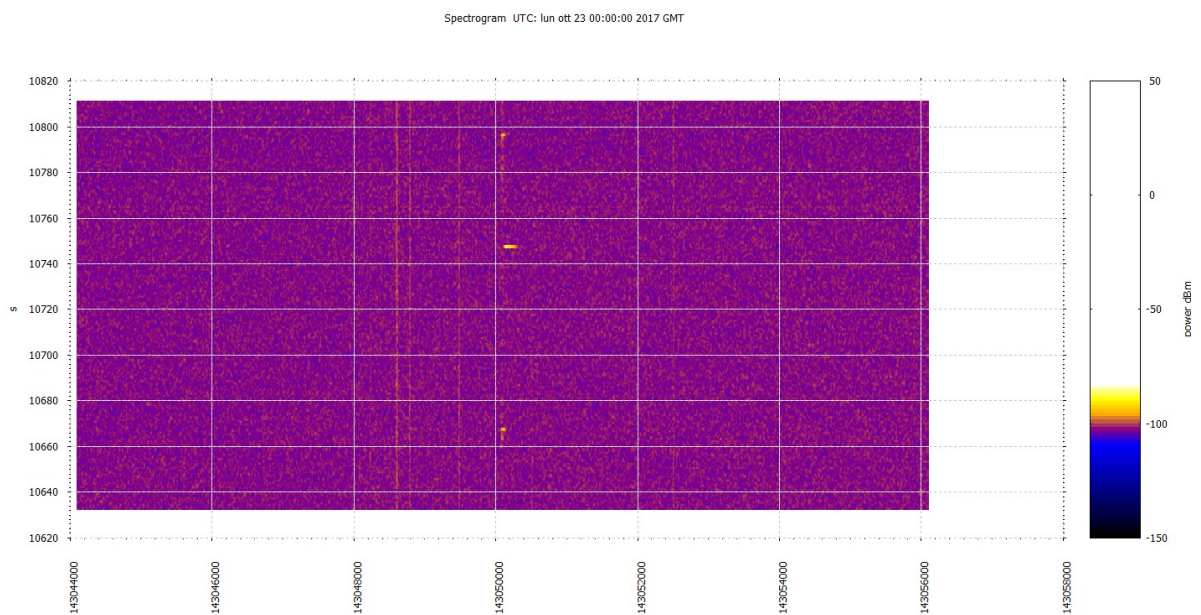
The command files are simple ASCII text files, so they can be edited later, to better suit user's needs. There are many examples at [gnuplot.info](http://gnuplot.info) site to learn how to personalize the plots.

When opening the command file with GNUplot, it will start a sequential presentation of all the plots generated during the session. A small dialog box appears to display the next plot, till the end. The box can be dragged in a corner with the mouse. The plotting window supports some keyboard/mouse commands to zoom / shift / rotate the plot. Plots can also be saved as image files.

The plot data are still ASCII files. The data files produced in periodic or automatic modes follow this naming convention:

`spectra_<configuration name>_<acquisition mode>_<date in ISO-8601 format>_<nnnnn>.dat`

where the last 5 digits are the event/screenshot reference number (column 1 in statistic CSV). In continuous mode, this number is not provided.



*Fig. 23: 2D color-mapped spectra of echoes captured from GRAVES radar.*

The following lines come from a plot data file, taken while running the Test Patterns. There is no difference in data format for 2D and 3D plots, they could be both plotted from the same data. It's only the GNUplot command file that makes the difference.

Each line represents the three informations recorded in a waterfall's pixel. The number of pixel recorded per scan is in fact limited to the peak detection range set (par.6.3.4) in order to keep the file compact.

The first column is the time in seconds, starting from the beginning of acquisition session (the absolute UTC date/time of session's beginning is coded in filename).



The second column is the frequency of the FFT point in Hz, (in Test pattern there is no dongle, so no tuner. The tune frequency always set as half of maximum sample rate: 1,6MHz ).

The third column is the power in dBfs.

37.100	2092897	-105.22				
37.100	2094318	-103.77				
37.100	2095738	-102.57				
37.100	2097159	-106.38				
37.100	2100000	-104.60	37.100	-104.61	-101.41	3.20 <--- last point
						<--- end of scan
						<--- starting the next scan
37.200	1100000	-105.05				
37.200	1101420	-104.32				
37.200	1102840	-102.98				
37.200	1104261	-105.25				

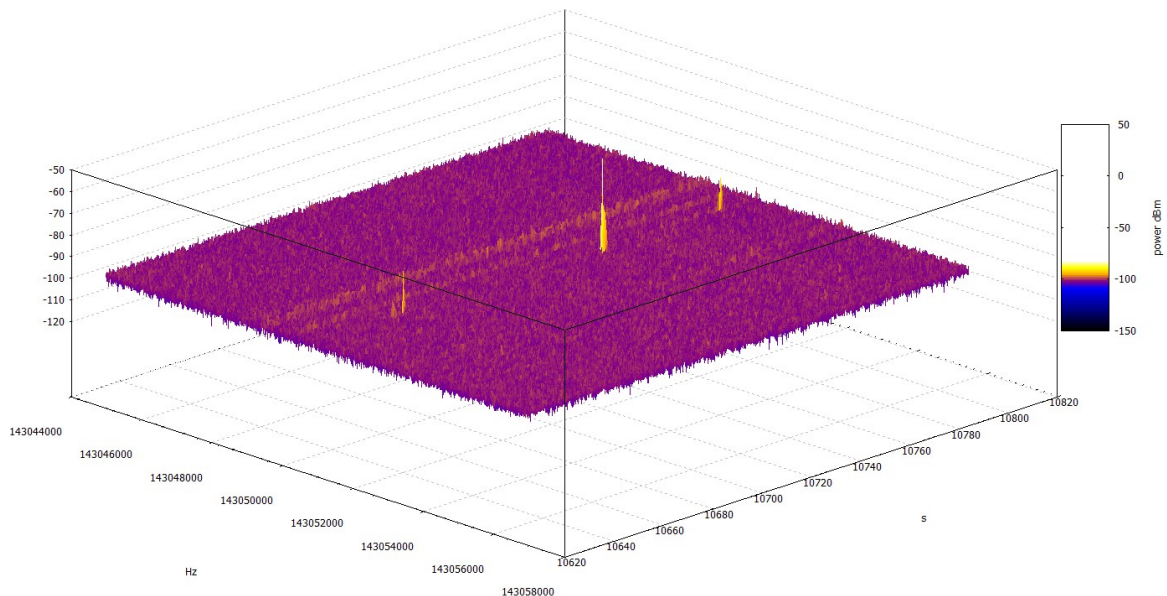
when each scan terminates, the following 4 columns are added to the last line:

The fourth column is a repetition of first one: time in seconds.

The fifth column is the average signal level in the scan (N).

The sixth column is the maximum signal level in the scan (S).

The seventh and last column is the difference (S-N) in the scan.



*Fig. 24: The same data showed in previous figure are now plotted in 3D*

Considering now the “2D total power” plot type, the data file produced consists only of rows containing the last 4 columns just described above. The total power plot is drawn based only on scan's statistic, not the full spectra data. This characteristic makes this format more attractive for continue registrations since the data file takes much smaller space on disk compared to full spectral data. The resulting plot looks similar to the “power history” graph at waterfall's right (par. 6.4.1.)

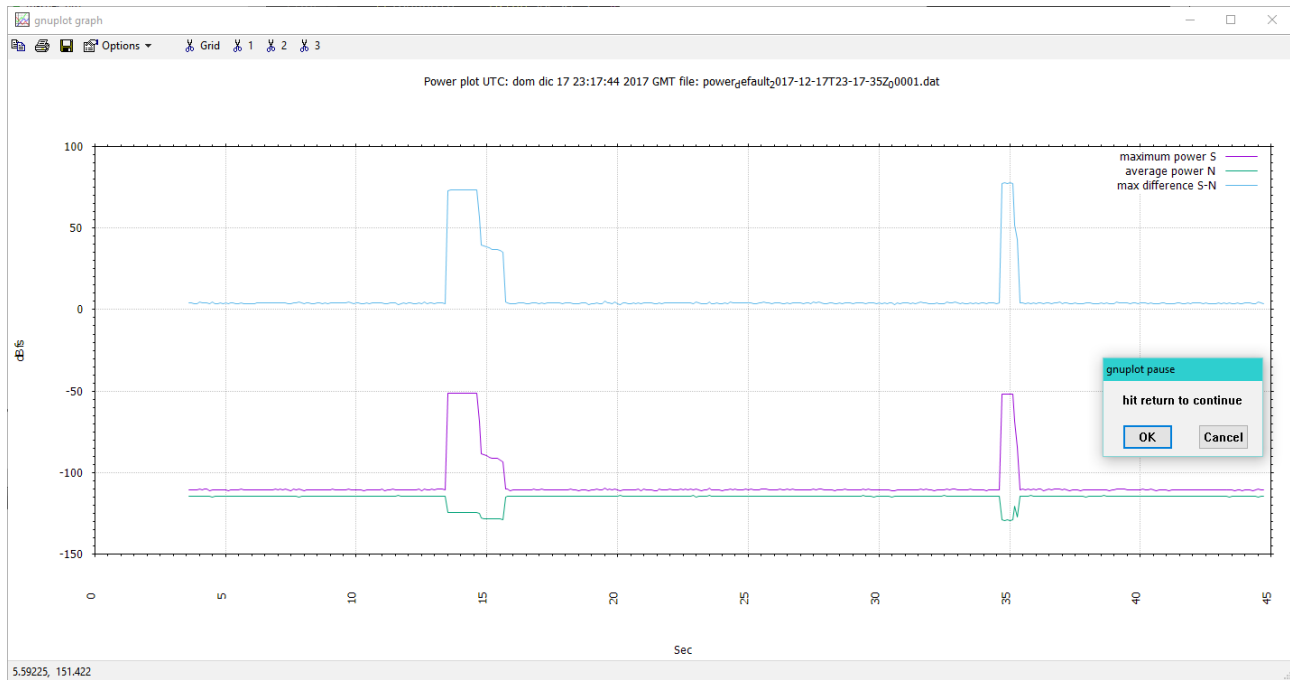


Fig. 25: 2D total power plot sample

The power data files produced in periodic or automatic modes follow this naming convention:

`power_<configuration name>_<acquisition mode>_<date in ISO-8601 format>_<nnnnn>.dat`

where the last 5 digits are the event/screenshot reference number (column 1 in statistic CSV). In continuous mode, that number is not provided.

### 7.4.6 Daily report

It's a short CSV table file that resumes the counts of all the captured events contained in the statistic CSV files found in working directory. There is one row per each day covered by those files and the events found are counted separately hour by hour and discriminated by lasting times in underdense, overdense and fakes.

Besides counts, two further columns contains the maximum signal S [dBm] and the average noise N, hour by hour.

Finally, the rightmost group of columns contains the daily totals.

These file are labeled as:

`daily_<configuration name>.csv`

The daily report file active is only one per configuration. It is created from scratch at midnight UTC each time a new configuration file has been loaded and run; since that day, new lines are appended, day by day.



	A	B	C	D	E	F	G	FF	FG	FH	FI	FJ	FK	FL	FN	FO	FP	FQ	FR	FS	FT	F
1	Echoes Daily Report																					
2	Date:	00h							23h							Daily totals						
3		Max S [dBm]	Avg N [dBm]	Total	overdenses	underdenses	fakes		Max S [dBm]	Avg N [dBm]	Total	overdenses	underdenses	fakes		Max S [dBm]	Avg N [dBm]	Total	overdenses	underdenses	fakes	
4	mar mag 1 2018	-	-	0	0	0	0		-110.65	-131.723	5	0	5	0		-	-	169	30	139	0	
5	mer mag 2 2018	-101.901	-131.276	7	0	7	0		-101.901	-130.599	11	1	10	0		-	-	153	12	141	0	
6	gio mag 3 2018	-989.257	-129.041	5	0	5	0		-870.339	-128.406	3	0	3	0		-	-	122	13	109	0	
7	ven mag 4 2018	-870.339	-128.515	2	0	2	0		-	-	0	0	0	0		-	-	198	22	176	0	
8	sab mag 5 2018	-	-	0	0	0	0		-928.481	-124.771	2	0	2	0		-	-	5	2	3	0	
9	dom mag 6 2018	-105.961	-125.112	1	0	1	0		-106.593	-126.675	5	1	4	0		-	-	132	37	95	0	
10	lun mag 7 2018	-984.671	-126.64	3	1	2	0		-100.543	-123.534	3	0	3	0		-	-	120	14	106	0	
11	mar mag 8 2018	-987.516	-120.651	3	0	3	0		-114.575	-132.227	3	1	2	0		-	-	115	12	103	0	
12																						
13																						

Fig. 26: Partial view of a daily report table

### 7.4.7 Full report

The full report is a dynamic html document that is created only by request in the working directory. It can be opened with a browser then saved as *pdf* to make it a more portable and compact document.

The full report file name follows this naming convention:

`<configuration name>-<start date>_to_<end date>.html`

The file content has been already described at par. 6.3.5

## 7.5. Daily archive

The working directory, in the previous Echoes releases, was a big “pot” filled with every file produced by the program. The unique form of management provided was the “Data lasting” function, running daily, to delete the files older than *n* days.

In my experience, after one month of acquisition the working directory contained about 50GB of data, where shots, plots and statistic tables were mixed together.

Starting from this release, at midnight UTC a 5-levels folders hierarchy is created (Fig. 27) under the working directory, where all the files produced in the past day are moved.

At first level is a folder having the same name of the configuration (*rts*) file running.

Under that, at second level, another folder is created each day, named as the past day. So, when “Data lasting” is *n*, the number of day folders at second level will be limited to *n*.

In the third level, under the day folder, along with the daily report csv table, three further folders are created. The first stores screenshots, the second stores plots and related command files while the third contains the statistic csv tables.

The fourth and fifth level folders are created only for shots and plots, to subdivide the related files by category: underdense, overdense and false positives. Actually the latter is always empty; *Echoes* didn't learned yet well how to recognize the false positives, the discrimination based only on the event lasting is too simple to be useful.

The plots are also subdivided by power and spectra. This was not really needed, since only one of the two plots can be produced during acquisition, but it has been done to prevent eventual future changes.

After archiving, the working directory contains only the configuration files and the program's log.

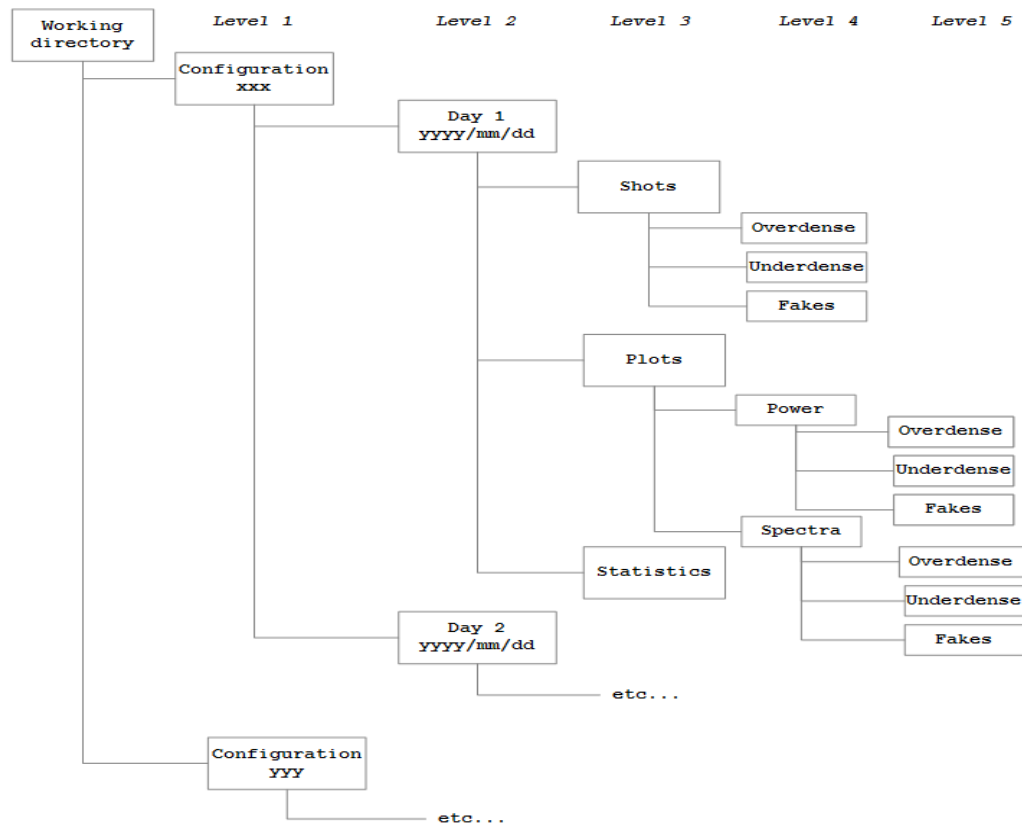


Fig. 27: Daily archive 4-levels folders structure

### WARNING:

The archive structure on the filesystem reflects the file *daily\_counters.ds* internal structure, that is placed in the *<Configuration xxx>* subfolder. If the file content does not match the archive structure (for instance, because some subfolders have been deleted by hand) the full report generation could fail, both automatic and manual.

## 7.6. Test patterns

When no RTL-SDR dongles are connected, the program starts asking to start playing the Test pattern. After pressing “Ok”, in the first tab (“Device”) of main window “Test pattern” is displayed under the “Device” control (Fig. 6) .

Test patterns are born mainly for development purposes, but they can be a simple way to test some program features even without dongles handy.

The patterns available are two. When starting in continuous mode, the function generator (yellow block in Fig. 2) produces a sweeping sine wave, resulting in the spectra shown in Fig. 13. The characteristics of this wave can be changed acting on sample rate (max frequency), gain (wave amplitude) and resolution (sweep step) controls (par. 6.3.2 and 6.3.3).

With periodic and automatic modes, the function generator roughly simulates meteor events, distanced by a pseudo-casual time interval between 100 to 600 times the refreshing interval.

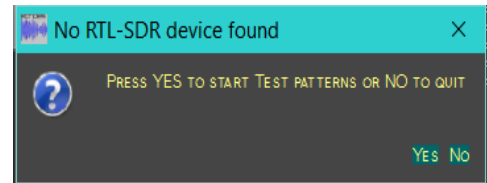


Fig. 28: Test patterns message box

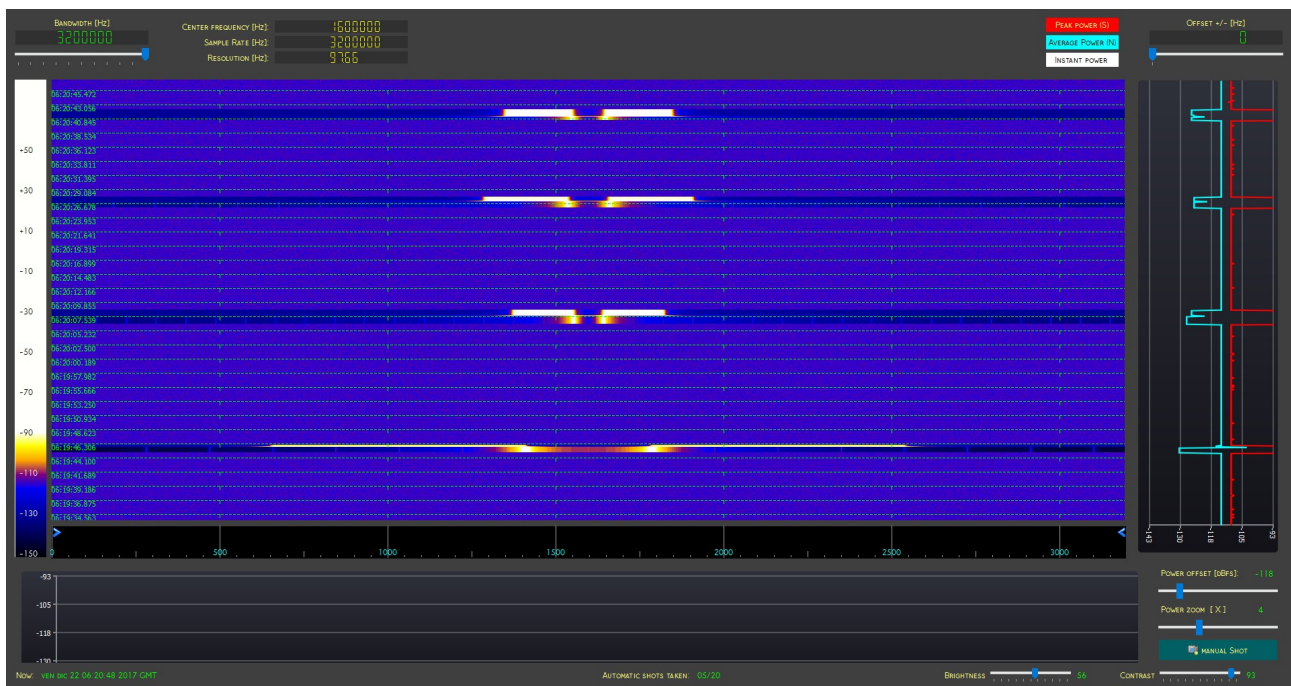


Fig. 29: test patterns in automatic mode

The doesn't look really like meteor echoes, but they're good anyway to check screenshots and plots generation, and even the report production without a RTLSDR dongle handy.