JAL: UHF spectrum analyzer

last modified: oct-2014, Stef Mientki

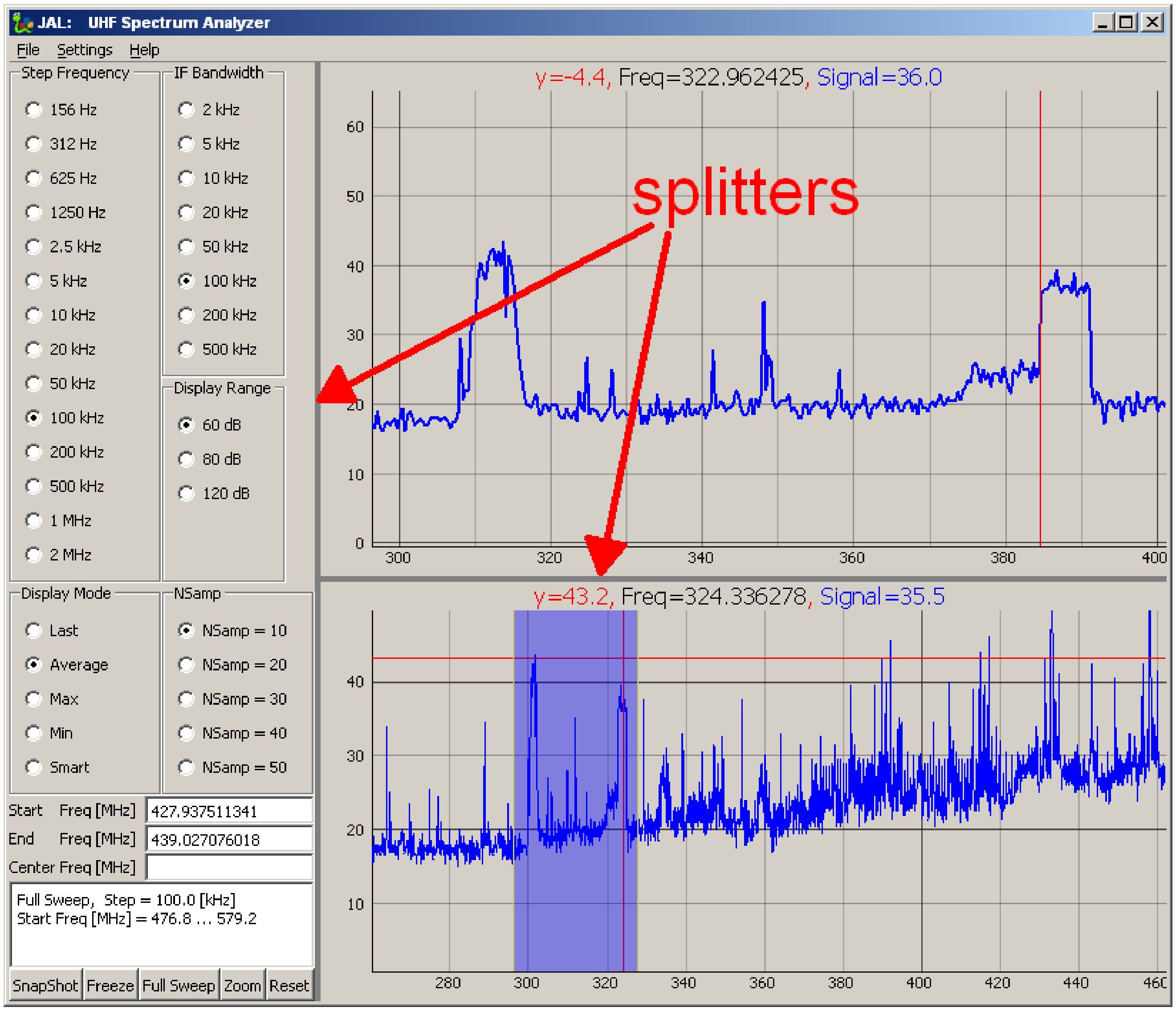
Introduction

This document describes a poor mans UHF spectrum analyzer based on a PIC (programmed in JAL) and a SI4432 chip (some others might also work). The radio frequency ranges from 240 MHz to 900 MHz, stepsize can be as small as 156 Hz and sensitivity should be better than -120 dBm. The module used is the 470M, which can be bought on ebay for about 3 €.

The 470M is a complete SPI module, containing theSI4432 transce□iver (), a 30 MHz crystal, an RF transmit and receive circuit, a antenna circuit (including antenna switcher G4C) and an antenna. The SI4432 has a programmable packet handler, supports FSK, GFSK and OOK modulation and 3 general purpose IO-pins. Temperature sensor, Battery Level detector and general purpose 8-bit ADC. One of the possibilities of the IO-pins is to generate a (programmable) clock for the PIC. Furthermore has many features to achieve an optimal power consumption.

The spectrum analyzer has the following feature:

- automatic comm port detection
- all settings directly available through the GUI
- scan mode, which scans the whole frequency range from 250 MHz to 900 MHz in steps of 100 kHz, Flexible zoom options, both static and life
- intelligent display modes (min, smart)
- search mode, which scan continuously the specified frequency range
- 5 spare buttons for experimental use



y=-4.4, Freq=322.962425, Signal=36.0

y=43.2, Freq=324.336278, Signal=35.5

Step Frequency: 156 Hz, 312 Hz, 625 Hz, 1250 Hz, 2.5 kHz, 5 kHz, 10 kHz, 20 kHz, 50 kHz, [✓ 100 kHz], 200 kHz, 500 kHz, 1 MHz, 2 MHz

IF Bandwidth: 2 kHz, 5 kHz, 10 kHz, 20 kHz, 50 kHz, [✓ 100 kHz], 200 kHz, 500 kHz

Display Range: [✓ 60 dB], 80 dB, 120 dB

Display Mode: Last, [✓ Average], Max, Min, Smart

NSamp: [✓ NSamp = 10], NSamp = 20, NSamp = 30, NSamp = 40, NSamp = 50

Start Freq [MHz] 427.937511341

End Freq [MHz] 439.027076018

Center Freq [MHz]

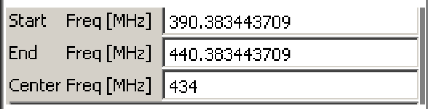
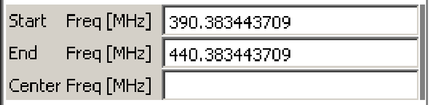
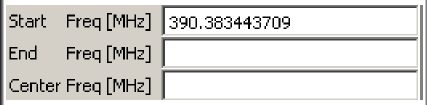
```
Full Sweep,  Step = 100.0 [kHz]
Start Freq [MHz] = 476.8 ... 579.2
```

Starting the program

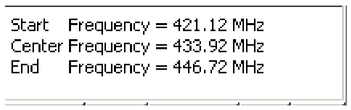
After connecting the PIC to a desktop computer, start the desktop program, which will search for the correct Comm-port. When the program has found the Comm-port, it will scan continuously the last selected frequency range and showing the result in the top window. The first time you use the program it will scan the range 420 .. 445 MHz, which includes 434 MHz, used in most domotic applications. F1-key shows this document, shift-F1 shows a Dutch frequency list.

Set Frequency range

In normal mode the selected frequency range is scannned in 256 steps. The frequency range can be set with one of the following methods (Enter-key in one of the Edit Fields will activate the frequency settings)

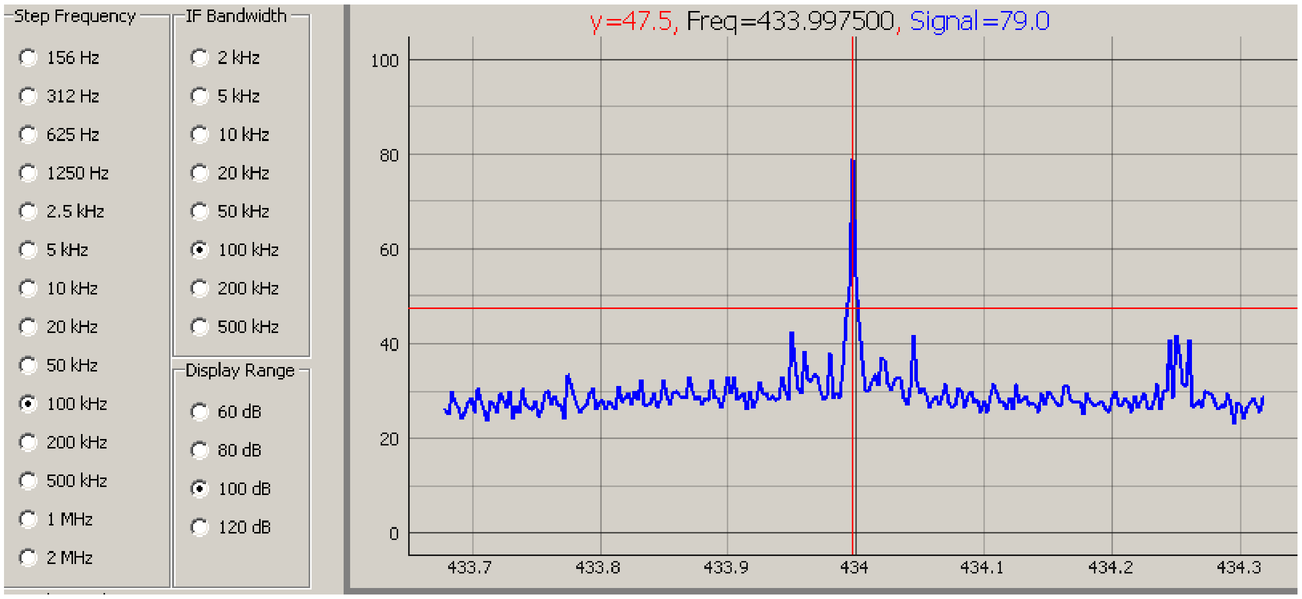
 Start Freq [MHz] 390.383443709 End Freq [MHz] 440.383443709 Center Freq [MHz] 434	**Center Frequency** is used. Step Frequency is taken from the radiobuttons. Start-Frequency is calculated from Start-Frequency = Center-Frequency - 128 * Frequency-Step. The nearest available frequency is used.
 Start Freq [MHz] 390.383443709 End Freq [MHz] 440.383443709 Center Freq [MHz]	**Start and End Frequency** are used. Step Frequency is calculated The selected Step-Frequency is used to scan the region. The End-Frequency is calculated from End-Frequency = Start-Frequency + 256 * Step-Frequency.
 Start Freq [MHz] 390.383443709 End Freq [MHz] Center Freq [MHz]	**Start Frequency** is used. Step Frequency is taken from the radio buttons. The Step-Frequency is calculated from (End-Frequency - Start-Frequency) / 256. The nearest available value for the Step-Frequency is used, resulting in a possible deviated End-Frequency.

The calculated values really used for the scan are always shown in the memo-filed at the left-bottom.



```
Start   Frequency = 421.12 MHz
Center Frequency = 433.92 MHz
End     Frequency = 446.72 MHz
```

If you want to zoom on a specific region the easiest way to do it, is to set the center frequency and then use the Step-Frequency to zoom in as shown below.

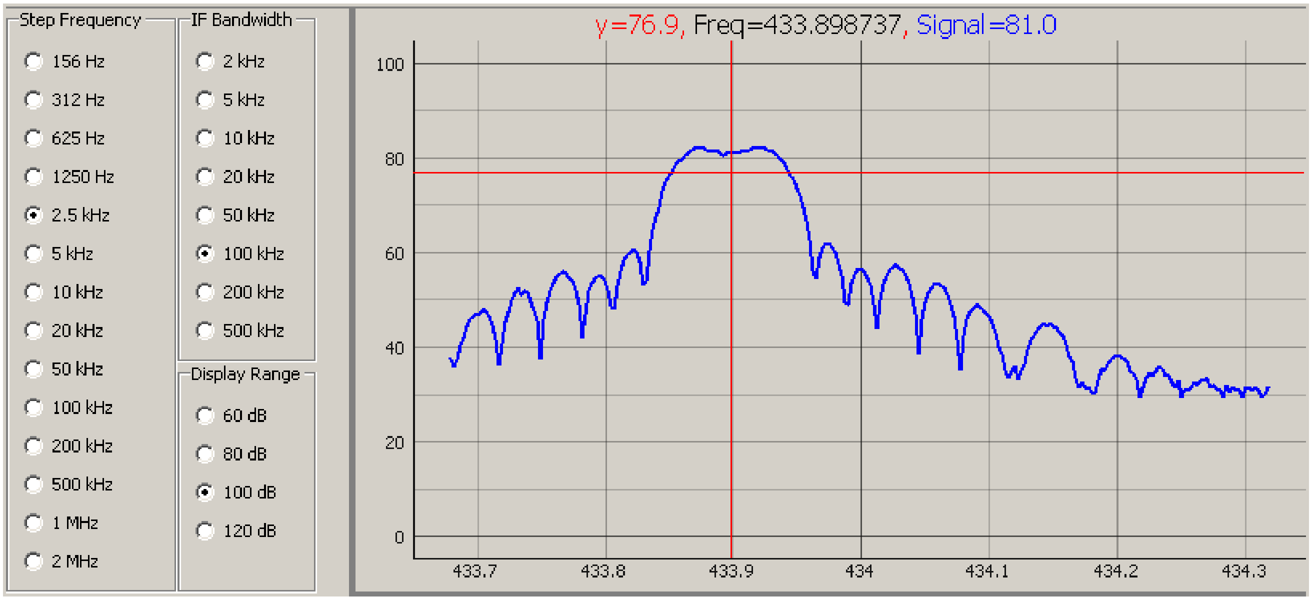


Step Frequency: 156 Hz, 312 Hz, 625 Hz, 1250 Hz, 2.5 kHz, 5 kHz, 10 kHz, 20 kHz, 50 kHz, [✓ 100 kHz], 200 kHz, 500 kHz, 1 MHz, 2 MHz

IF Bandwidth: 2 kHz, 5 kHz, 10 kHz, 20 kHz, 50 kHz, [✓ 100 kHz], 200 kHz, 500 kHz

Display Range: 60 dB, 80 dB, [✓ 100 dB], 120 dB

y=47.5, Freq=433.997500, Signal=79.0



Step Frequency: 156 Hz, 312 Hz, 625 Hz, 1250 Hz, [✓ 2.5 kHz], 5 kHz, 10 kHz, 20 kHz, 50 kHz, 100 kHz, 200 kHz, 500 kHz, 1 MHz, 2 MHz

IF Bandwidth: 2 kHz, 5 kHz, 10 kHz, 20 kHz, 50 kHz, [✓ 100 kHz], 200 kHz, 500 kHz

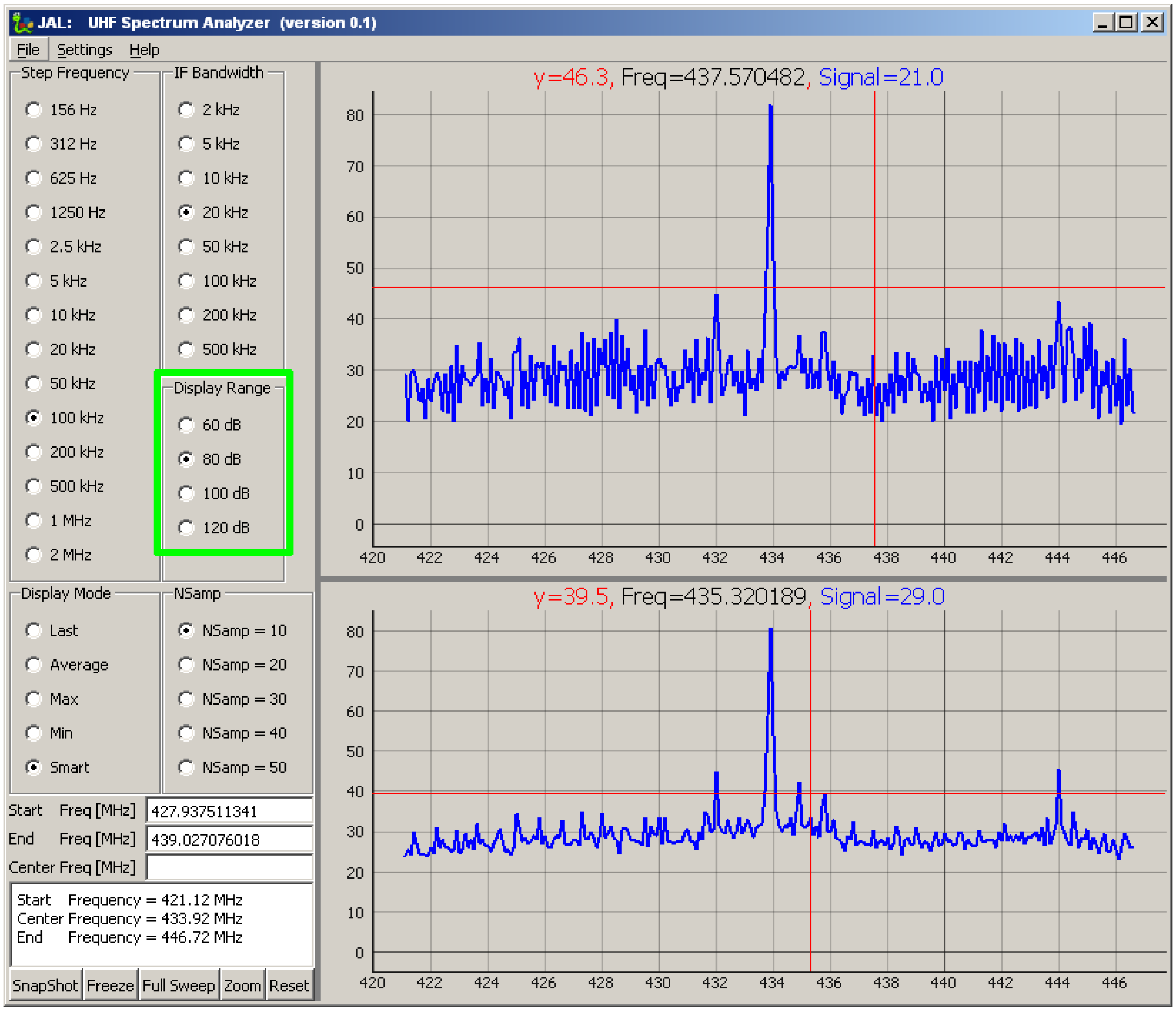
Display Range: 60 dB, 80 dB, [✓ 100 dB], 120 dB

y=76.9, Freq=433.898737, Signal=81.0

Snapshot / Freeze

The Freeze button toggles the top window between Life updating and Freezed display.
Either in Freeze-Mode or Life-Mode you can take a snapshot, which will copy the top window to the bottom window. In this way you can compare the spectrum changes over time, compare different settings and/or different frequency ranges. Below is an example shown with two different settings of the display mode (later more about that).

Hoovering with the mouse in a frozen graph (bottom window always, top window in Freeze mode) displays at the top of each window the values of the curve at the crosshair: black is Frequency, blue is the signal value, red is the value of the horizontal line of the crosshair.


JAL: UHF Spectrum Analyzer (version 0.1)

File Settings Help

Step Frequency: 156 Hz, 312 Hz, 625 Hz, 1250 Hz, 2.5 kHz, 5 kHz, 10 kHz, 20 kHz, 50 kHz, [✓ 100 kHz], 200 kHz, 500 kHz, 1 MHz, 2 MHz

IF Bandwidth: 2 kHz, 5 kHz, 10 kHz, [✓ 20 kHz], 50 kHz, 100 kHz, 200 kHz, 500 kHz

Display Range: 60 dB, [✓ 80 dB], 100 dB, 120 dB

Display Mode: Last, Average, Max, Min, [✓ Smart]

NSamp: [✓ NSamp = 10], NSamp = 20, NSamp = 30, NSamp = 40, NSamp = 50

Start Freq [MHz] 427.937511341
End Freq [MHz] 439.027076018
Center Freq [MHz]

```
Start  Frequency = 421.12 MHz
Center Frequency = 433.92 MHz
End    Frequency = 446.72 MHz
```

y=46.3, Freq=437.570482, Signal=21.0

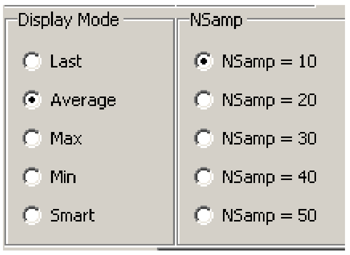
y=39.5, Freq=435.320189, Signal=29.0

The Y-axis scaling of both the plots are set by the "Display Range", the green framework in the above picture.

The Reset button will reset the PIC (I never used it until now). After resetting the PIC, the buttons of the desktop program and PIC-settings will in general be out of sync.

Nsamp / Display-Mode

With the setting of NSamp and Display-Mode a choice for optimal noise cancelation can be found, depending on the nature of the signal and the type of noise present. The selection of NSamp is often not so obvious and in fact we would like to make NSamp much longer (noise reduction is the square root of the number of samples) but speed of scanning is too much affected. So it's best to leave the Nsamp = 10.


Display Mode: Last, [✓ Average], Max, Min, Smart

NSamp: [✓ NSamp = 10], NSamp = 20, NSamp = 30, NSamp = 40, NSamp = 50

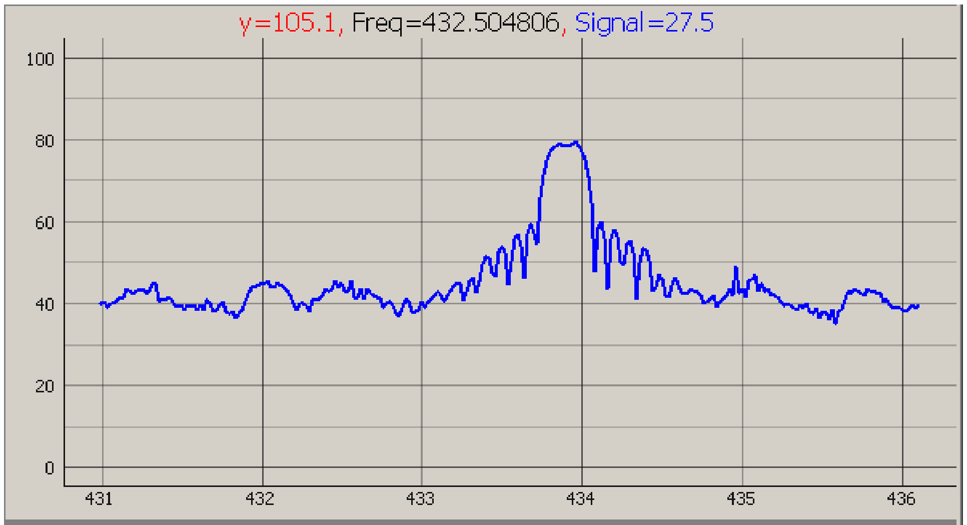
The Display Mode setting is a more interesting setting, and although not optimal yet can reveal some fine details in the spectrum. Let's first explain the different modes

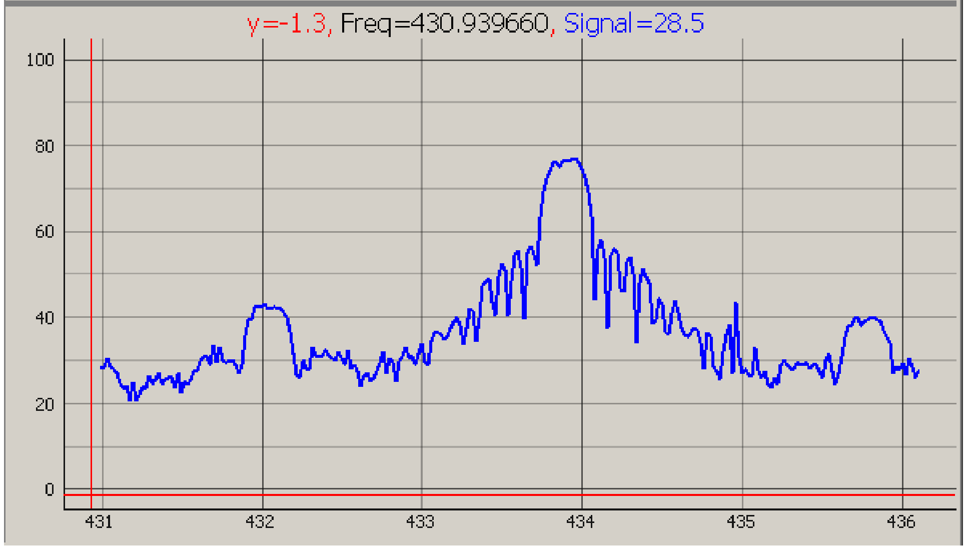
1. Last: in each frequncy bin that will be measured a sample (the last one) is taken

2. Average: the average power in each bin is displayed. This sounds good, but due to the short measuring time (10 .. 50 samples) and the nature of the noise, in most cases this doesn't give much improvement.
3. Max: the maximum value in each bin is displayed. In most cases no improvement.
4. Min: the minimum value in each bin is taken. Although it might sound strange, this often gives a huge improvement. The explanation is that if the bin contains noise it will detect the lowest power. In case there's a real signal, the noise doesn't affect the measured signal (logaritmic, dBm) so the total signal to moise ratio will improve.
5. Smart: this measurement shows when the signal is noisy. Although the algorithm is not correctly implemented yet, it sometimes works, see the example below. The smart mode is derived from the sweeping spectrum analyzer. If the sweep begins to detect a signal it will be low but will rise until the sweep frequency has reached the found signal frequency (due to the bandwidth-time limitation in fact a little bit later). So the idea is that if there's a real signal the detected level will continuously rise. If noise is detected the detected level will rise and fall. If the algorithm detecets a signal is will display the maximum detected value. If noise is detected the display will alternate between displaying min and max value so you'll get an impression of the amplitude of the noise.

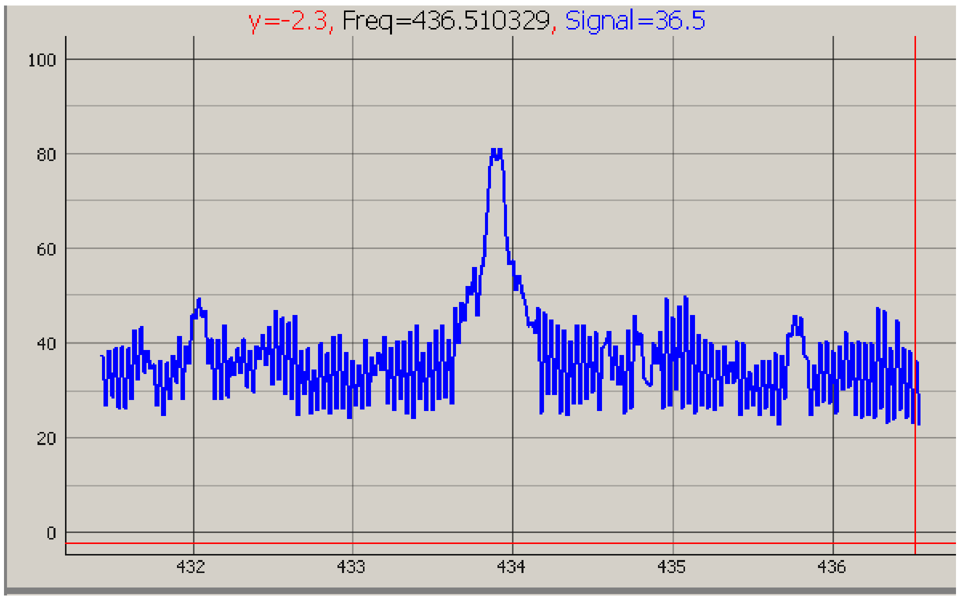
So in general it's best to try the Average-Mode or Min-Mode.

Here are two pictures of the same signal:
The top window the Average-Display-Mode is used: we only see a clear signal at 432.9 MHz
In the bottom window the Min-Display-Mode is used and we can clearly see that there are signals at 432 MHz and 435.7 MHz.





Example of the Smart Display Mode,
there's a real signal around 433.9 MHz an probably real signals around 432 MHz and 435.7 MHz



IF-Bandwidth

The IF-Bandwidth determines Resolution Bandwidth (RBW). This is the most diffult part to explain, so I'll leave that to others.

$$ST = \frac{k(Span)}{RBW^2}$$

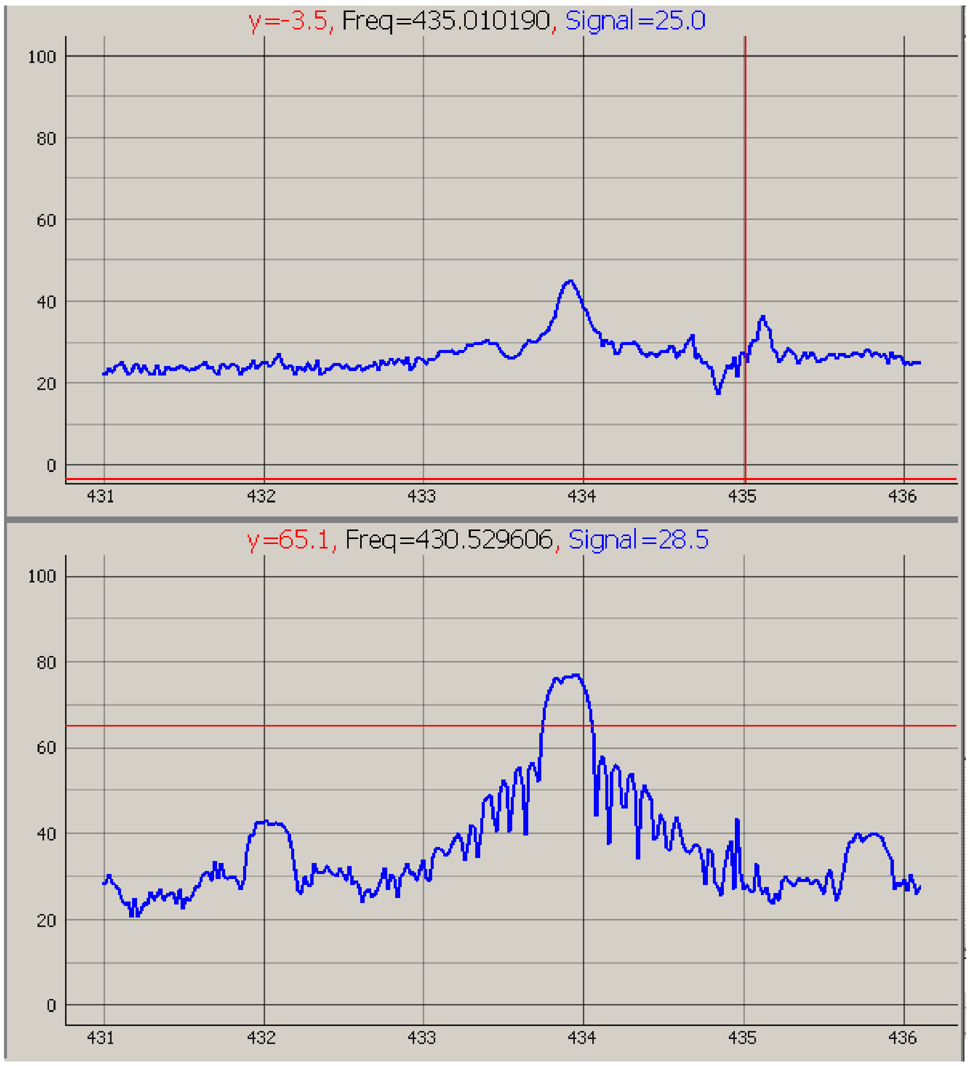
Just a few hints:

- In general setting the IF-Bandwidth to 100 kHz is fine for most measurements
- In general it's better to choose a higher IF-Bandwidth, better detection but higher noise level
- The IF-Bandwidth should in general have at least the value of the Step-Frequency
- The smaller the IF-Bandwidth is choozen the longer you've to measure to catch a signal (increase NSamp)

Some more explanation and some nice animations can be found on Wikipedia
http://en.wikipedia.org/wiki/Spectrum_analyzer

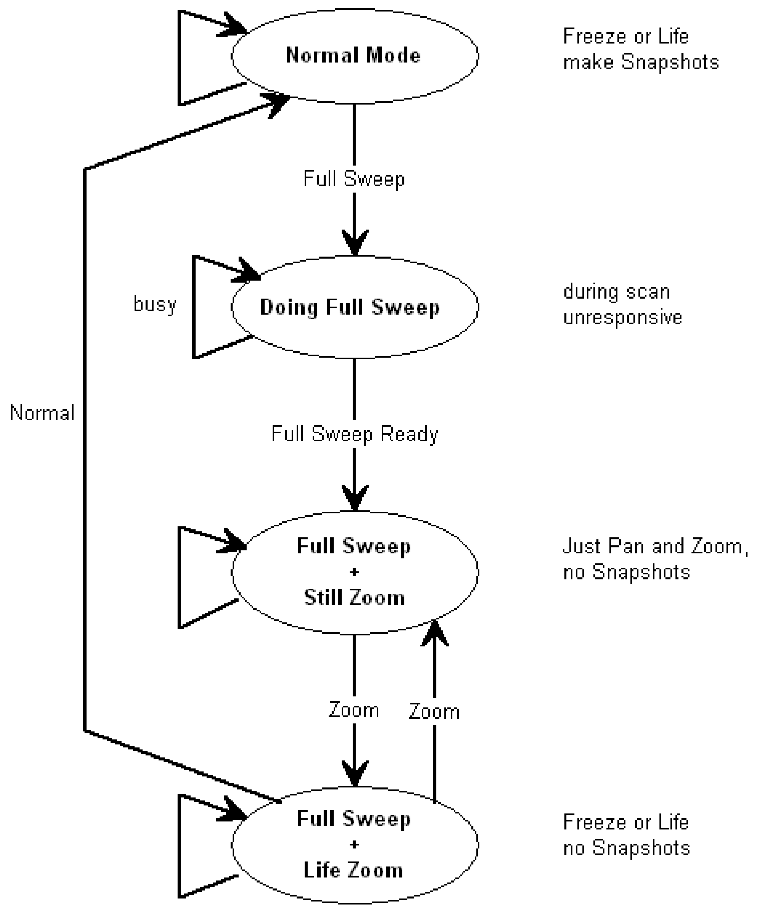
or this presentation from HP might give some more understanding
http://mientki.ruhosting.nl/data_www/raspberry/doc/hp_verhaal_5965-7920E.pdf

Here an example of choosing a too low IF-Bandwidth
In the bottom window the real signals can be seen well (IF-Bandwidth 100 kHz).
In the top window the same frequency band is scanned but with an IF-Bandwidth of 5 kHz. The signals at 432 MHz and 435.7 MHz are totally vanished.

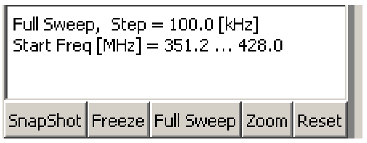


Global modes / Full Scan

You can make a full scan over the whole spectrum from 250 MHz to 900 MHz with a frequency resolution of 100 kHz. After the scan is made you can zoom into the recording or even look at life parts of the total scan. The spectrum analyzer is globally in one of the states shown in the state-diagram below.

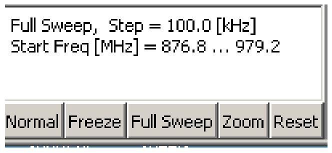


In all the paragraphs above the program was in the Normal Mode.
Pressing the button "Full Sweep" which will start the full sweep and takes a about 5 minutes. During the scan the program is unresponsive, but you can follow the progress in the memo at the left-bottom.



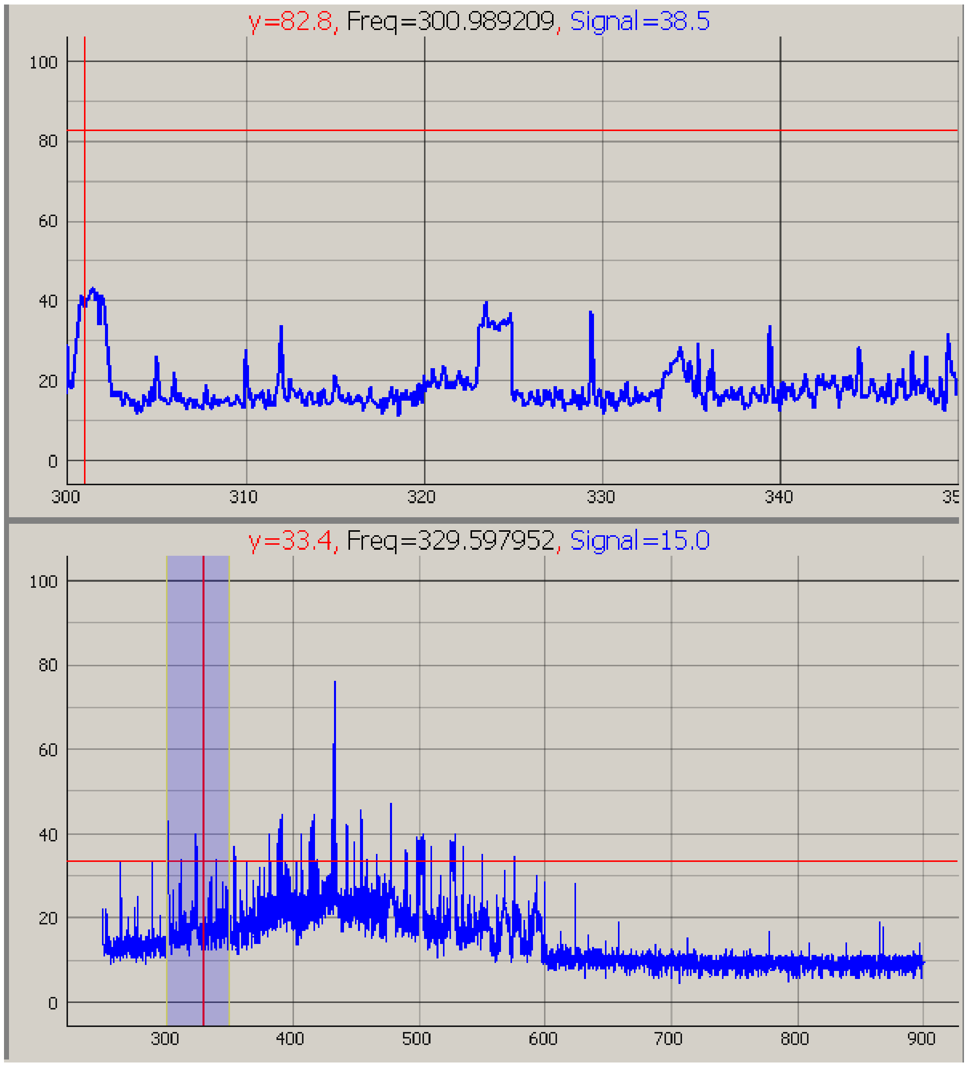
```
Full Sweep,  Step = 100.0 [kHz]
Start Freq [MHz] = 351.2 ... 428.0
```

When the scan is done, the Snapshot button will have another text (and meaning) acoording to the above state-diagram



```
Full Sweep,  Step = 100.0 [kHz]
Start Freq [MHz] = 876.8 ... 979.2
```

and you get a picture like this:


y=82.8, Freq=300.989209, Signal=38.5
100
80
60
40
20
0
300
310
320
330
340
35
y=33.4, Freq=329.597952, Signal=15.0
100
80
60
40
20
0
300
400
500
600
700
800
900

In the bottom window the total scan can be seen. The blue block is the zoomed region of the signal that is displayed in the top-window. The blue block can be dragged to position the zoom window. If you drag the left or right border of the blue block, this will resize the zommed region. You can also drag the topwindow (much better control because it's zoomed) in which case the blue block in the lower part will synchronuously move.

By pressing the Life button, the top-window will show the selected region in life, which gives you a great opportunity to see the selected region in far more detail and/or with different settings. Pressing Zoom again, changes the top-window back to the view that was recorded during the full-scan.

With the button Normal you can return to the Normal mode. The total scan in the bottom window will remain until a Snapshot is made or a new full-sweep is performed.

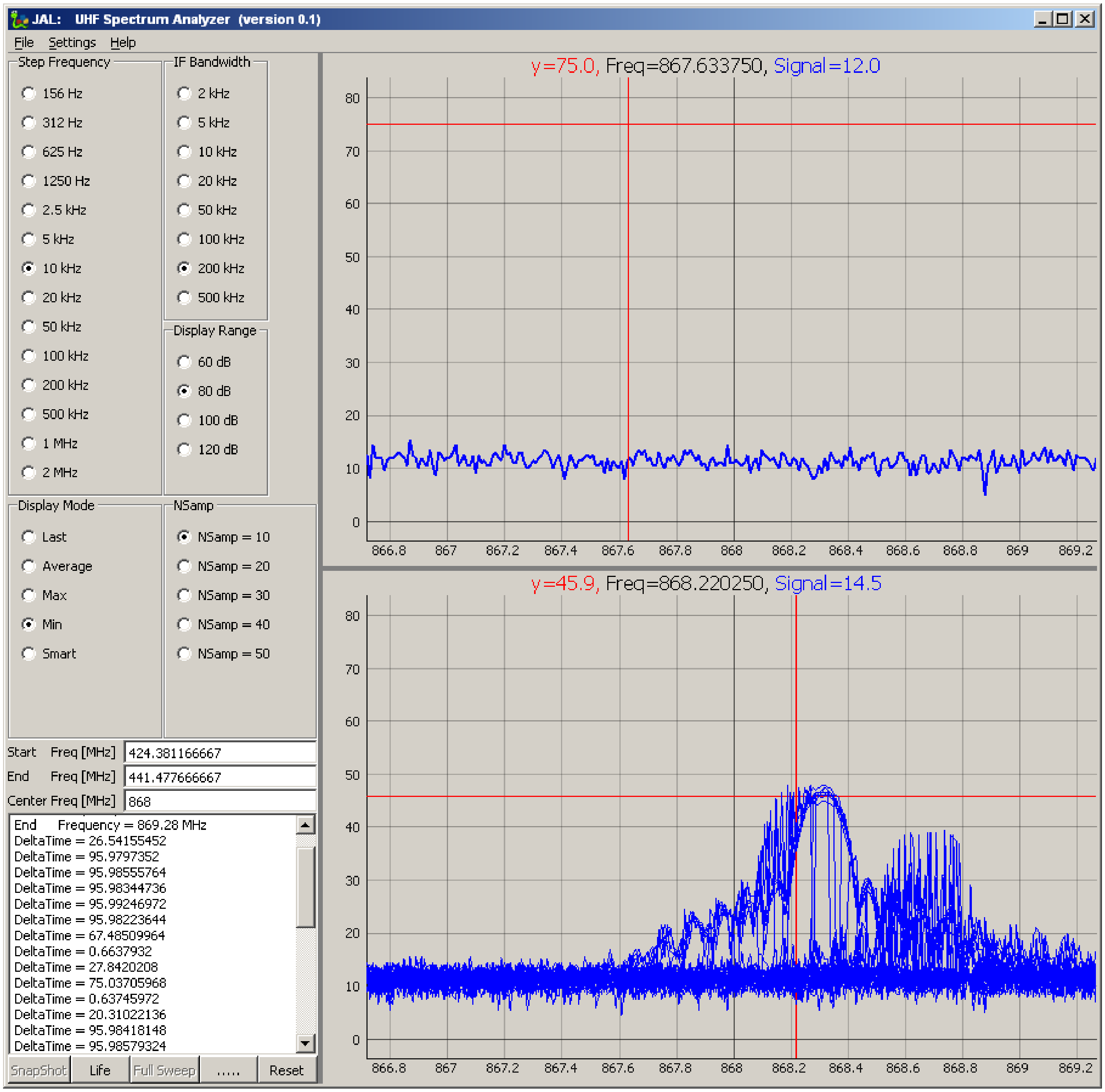
technical implementation

Mode	Freeze	Sweep	Snapshot
Normal	possible	0	yes
Busy Full Sweep	no	1	no
Full Sweep Ready	no	2	no
Full Sweep + Life Zoom	possible	3	no

Search

In the Search mode, the program continuously scans the specified range looking for possible interesting signals. The first scan is used a s a reference scan from which the average is used as a treshold. If a signal larger than 10 dB above the treshold is detected, the scan will be added to the bottom window and the time from the last detection is added to the memo.

In the picture below you can see that there are 2 signals, one centered 868.3 MHz and one around 868.7 MHz. From the memo you can see that one of the signals has a repetition rate of 96 seconds, which comes from my weather station (WS3000). The other signal is probably a weather station of one of my neighboors.



Step Frequency: 156 Hz, 312 Hz, 625 Hz, 1250 Hz, 2.5 kHz, 5 kHz, [✓ 10 kHz], 20 kHz, 50 kHz, 100 kHz, 200 kHz, 500 kHz, 1 MHz, 2 MHz

IF Bandwidth: 2 kHz, 5 kHz, 10 kHz, 20 kHz, 50 kHz, 100 kHz, [✓ 200 kHz], 500 kHz

Display Range: 60 dB, [✓ 80 dB], 100 dB, 120 dB

Display Mode: Last, Average, Max, [✓ Min], Smart

NSamp: [✓ NSamp = 10], NSamp = 20, NSamp = 30, NSamp = 40, NSamp = 50

Start Freq [MHz] 424.381166667

End Freq [MHz] 441.477666667

Center Freq [MHz] 868

```
End    Frequency = 869.28 MHz
DeltaTime = 26.54155452
DeltaTime = 95.9797352
DeltaTime = 95.98555764
DeltaTime = 95.98344736
DeltaTime = 95.99246972
DeltaTime = 95.98223644
DeltaTime = 67.48509964
DeltaTime = 0.6637932
DeltaTime = 27.8420208
DeltaTime = 75.03705968
DeltaTime = 0.63745972
DeltaTime = 20.31022136
DeltaTime = 95.98418148
DeltaTime = 95.98579324
```

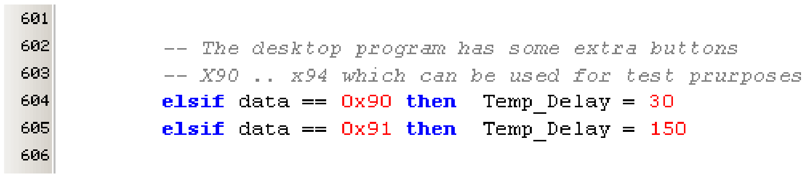
technical implementation

Mode	Freeze	Sweep	Snapshot
Normal	possible	0	yes
Busy Full Sweep	no	1	no
Full Sweep Ready	no	2	no
Full Sweep + Life Zoom	possible	3	no

General purpose buttons

There are 5 general purpose buttons, which sends commands with codes 0x90 .. 0x94 to the PIC.

In the JAL library SI4432_support you can easily attach some (experimental) extra functionality.



```

        -- The desktop program has some extra buttons
        -- X90 .. x94 which can be used for test prurposes
        elsif data == 0x90 then  Temp_Delay = 30
        elsif data == 0x91 then  Temp_Delay = 150

```

Ether Frequencies

http://mientki.ruhosting.nl/data_www/raspberry/doc/Frequenties%20Nederland%20totaal.pdf
http://radio-tv-nederland.nl/dvbt/digitenne-kpntv.html
http://radio-tv-nederland.nl/fmkaart/zenderkaarten-overzicht.html
http://appl.agentschaptelecom.nl/dav/index.html

History

november 2014: **version 0.3**

-
- adapted to the extended JAL library
- Time Sweep added (Nsamp slightly changed)
- Delay corrected proportional to 1/f, especially for small IF-Bandwidth (IF of 2 kHz still doesn't seem to work)

october 2014: **version 0.2**

- resize of zoom region now works
- added 5 general purpose buttons, sending 0x90 .. 0x94 to the PIC
- fixed bug: graphs where not in the right container
- antenna switch explictly set to receive mode

october 2014: **Version 0.1**, initial release

ToDo / Future ideas

- better Smart display mode
- AFC/ gain / Antenna switching / etc
- delta step overflow (in some rare cases)
- coupling with life frequency database
- weird behaviour (not explained by delta step overflow) e.g. steps in first picture

What you need

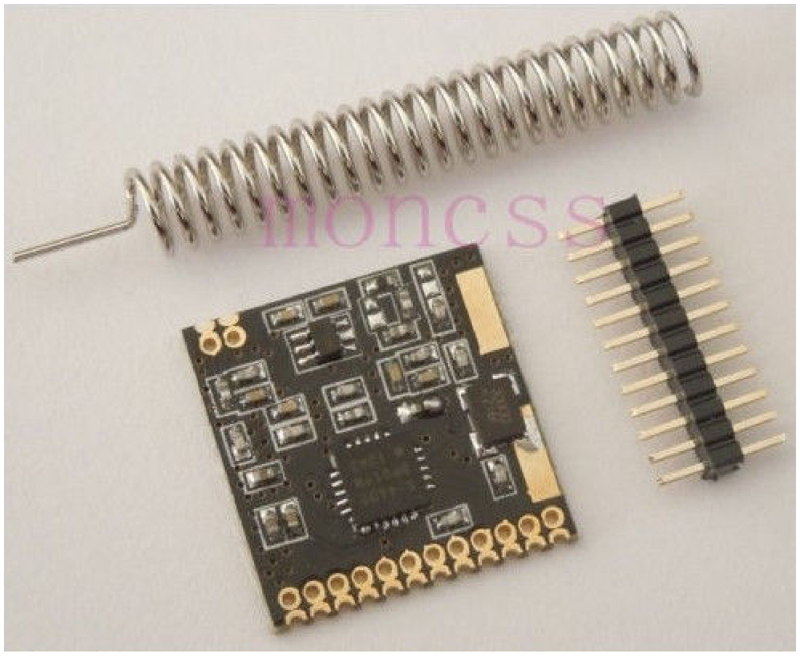
- PIC with SPI and Serial port (at the end of this paragraph the memory requirements for the PIC are given)
- wireless module 470M (RFM22b probably also works)
- JAL-program,...link....
- Python program
- Python compiler
- Computer with a serial port

PIC Memory requirements:

```
Code area: 5382 of 65536 used (bytes)
Data area: 773 of 3840 used
Software stack available: 3029 bytes
Hardware stack depth 3 of 31
```

The tranceiver 470M

Be aware that the pin spacing of this module is small, 0.05 inch (where 0.1 inch is the standard). The module has a SI4432 tranceiver, Xtal and a receive and transmit circuit. The board runs at 3V3.



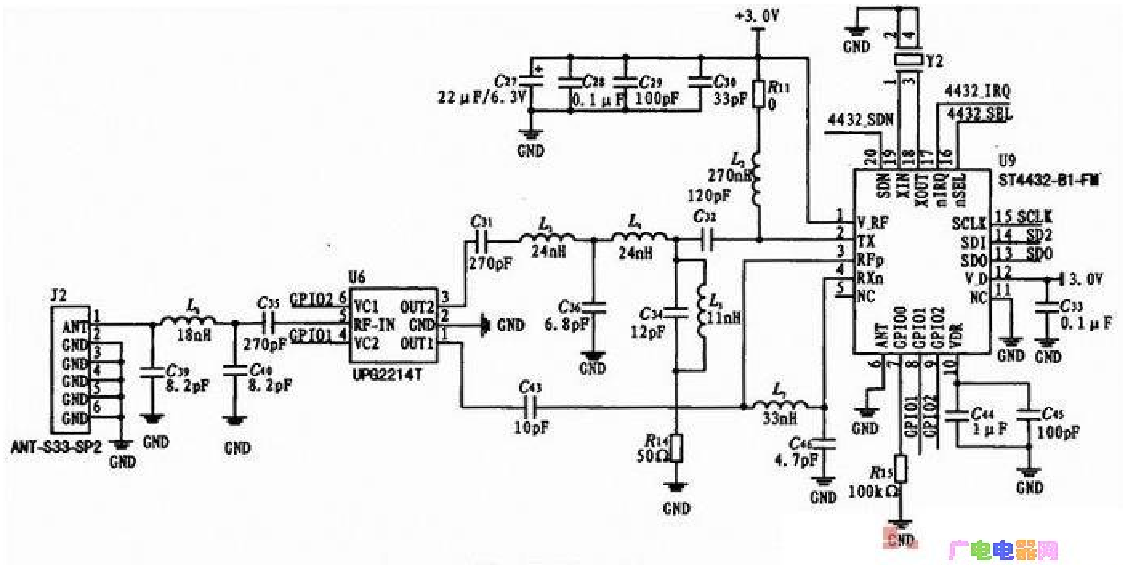
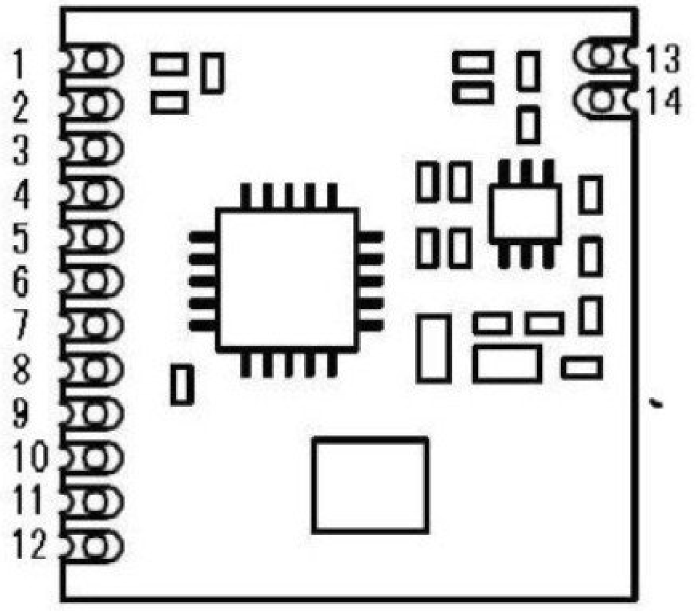
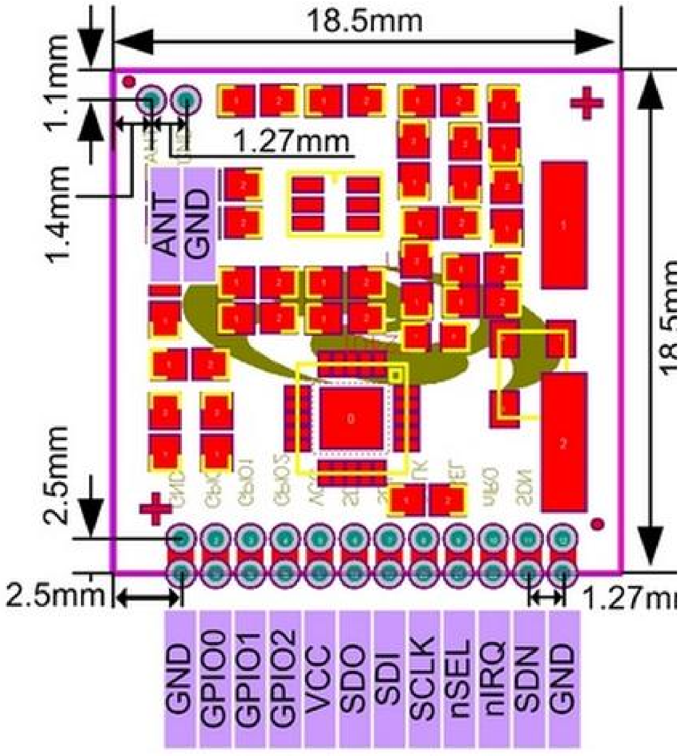


图 2 数据收发电路

The GPIO-pins to the antenna-switch are not correctly shown in the above figure. In fact the antenna switch is controlled by GPIO-0 and GPIO-1 (drive capability of the port is not important and can be left to 00).

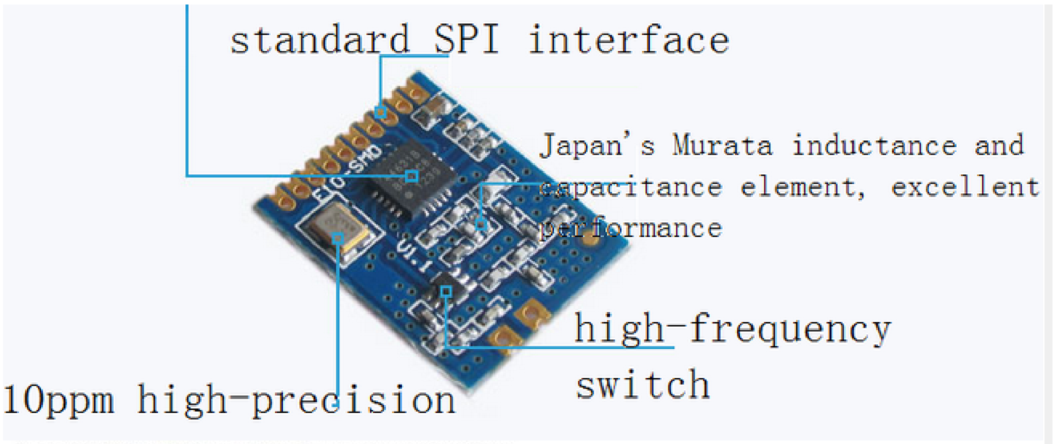


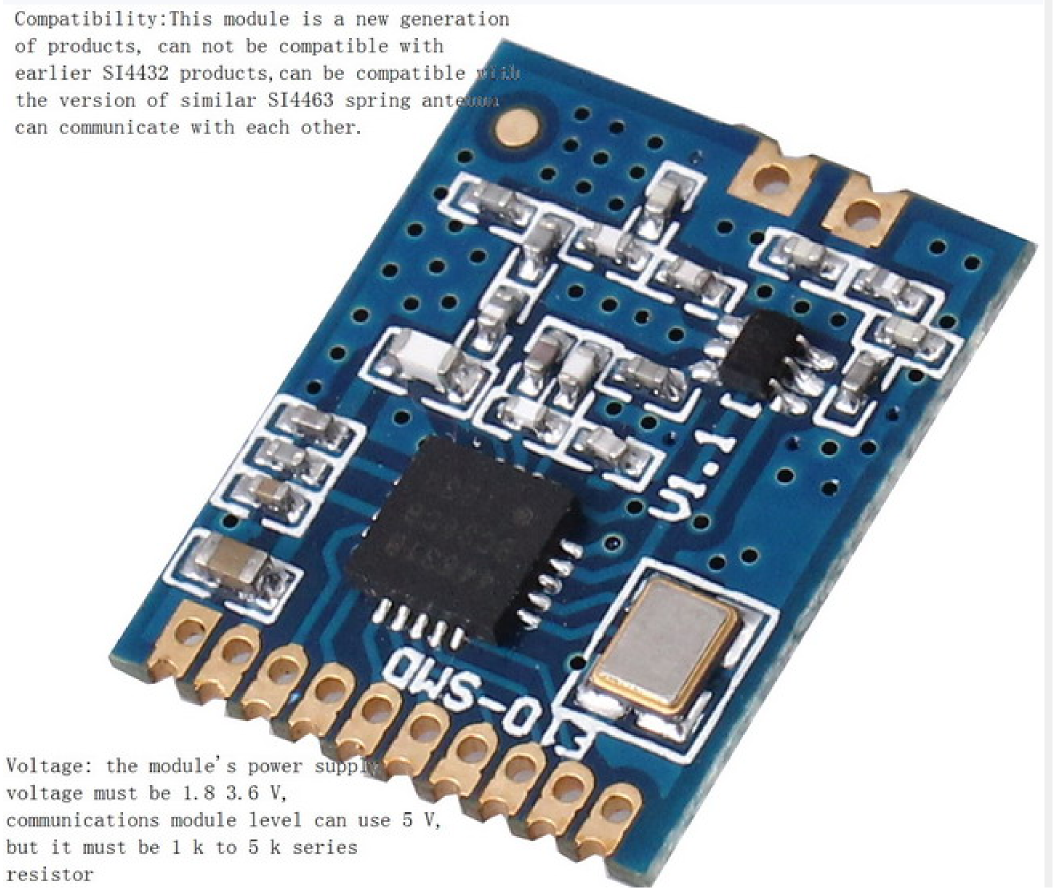


1 CND power supply
2 GPIO0 internal module, launch control foot
3 GPI01 internal access module to accept control foot
4 GPIO2 directly connected to the chip GIIO2 pin
5 VCC positive supply 3.3V
6 SDO 0-VDD V digital output provides a serial readback function of the internal control register
7 SDI serial data input. 0-VDD V digital input, this pin for 4-wire serial data clock function
8 SCLK serial clock input. 0-VDDV last month set of inputs. This pin provides a 4-wire serial data clock function
9 NSEL serial interface select input pin,0-VDDV digital input. This pin to 4-wire serial data bus select / enable function, this signal is a read / write mode.
10 NIRQ interrupt output pin
11 SDN Close the input pin. Input 0-VDDV digital. SDN = 0 in shutdown mode, so mode
12 GND access power ground
13 ANT 50 ohm coaxial antenna

Datasheet: http://www.dorji.com/docs/data/DRF4432F20.pdf

a slightly newer version, showing the HF-antenna switch


standard SPI interface
Japan's Murata inductance and capacitance element, excellent performance
high-frequency switch
10ppm high-precision

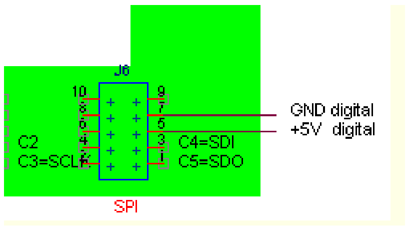


Compatibility:This module is a new generation of products, can not be compatible with earlier SI4432 products,can be compatible with the version of similar SI4463 spring antenna can communicate with each other.

Voltage: the module's power supply voltage must be 1.8 3.6 V, communications module level can use 5 V, but it must be 1 k to 5 k series resistor

The datasheet of the RF-antenna switch can be found here

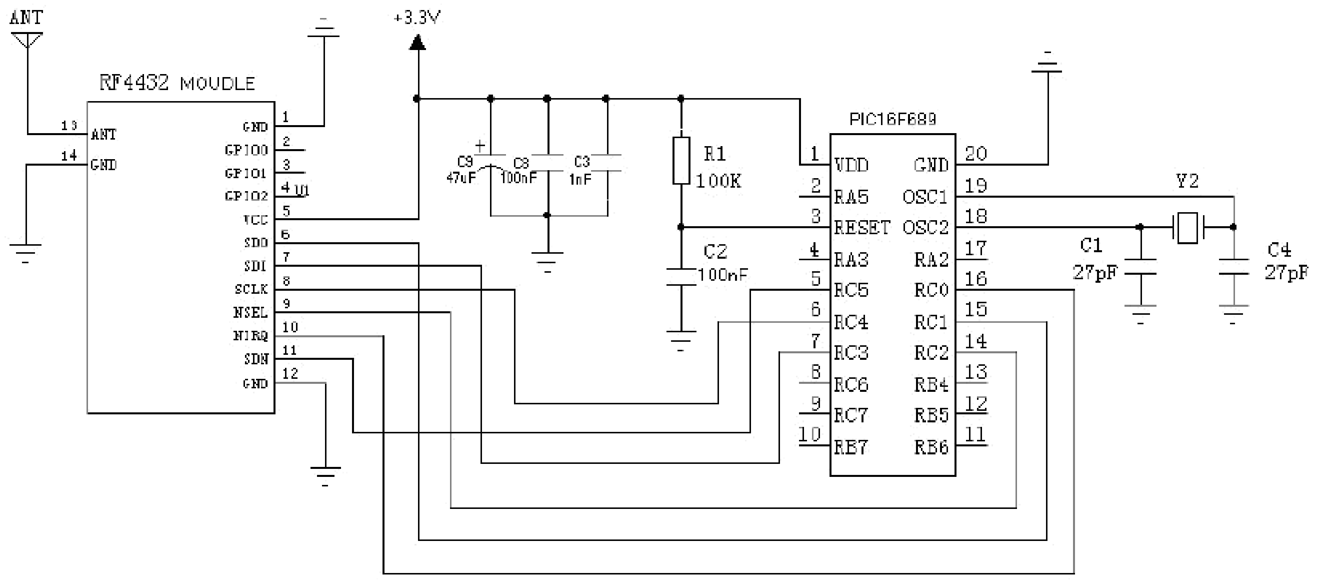
Connecting the PIC



The C-pin connections in the schematic below are not the standard ones for hardware SPI, see the main JAL program for further details.

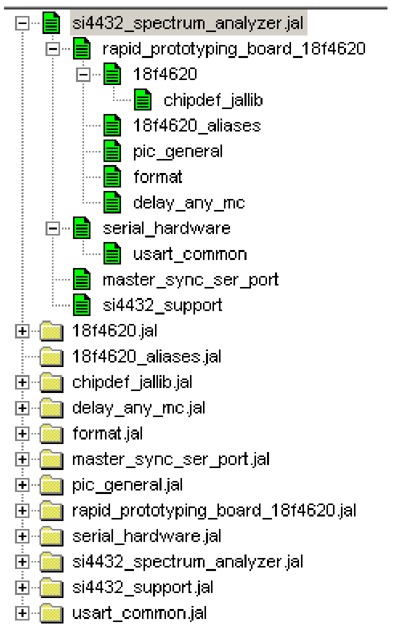
C2 = CEnable
C3 = SPI clock
alias MISO is pin_c4

alias MOSI is pin_c5



The JAL Program

The Jal program consists of the following files:



```
si4432_spectrum_analyzer.jal
    rapid_prototyping_board_18f4620
        18f4620
            chipdef_jallib
        18f4620_aliases
        pic_general
        format
        delay_any_mc
    serial_hardware
        usart_common
    master_sync_ser_port
    si4432_support
18f4620.jal
18f4620_aliases.jal
chipdef_jallib.jal
delay_any_mc.jal
format.jal
master_sync_ser_port.jal
pic_general.jal
rapid_prototyping_board_18f4620.jal
serial_hardware.jal
si4432_spectrum_analyzer.jal
si4432_support.jal
usart_common.jal
```

For this project we only wrote the main program SI4432_spectrum_Analyzer and the SI4432_support library.
A zip file containing all the above JAL files can be found here:

http://mientki.ruhosting.nl/data_www/raspberry/Spectrum_Analyzer/SI4432.zip

The pic definition files should ofcourse match the PIC you use.

The desktop program

The desktop program is a Python script that can be run if you've installed Python and the necessary standard libraries,
A zip file containing the self written Python files can be found here

http://mientki.ruhosting.nl/data_www/raspberry/Spectrum_Analyzer/UHF_Spectrum_Analyzer.zip

If you've Python (many Linux system already have Python installed), you can just run the main program UHF_Spectrum_Analyzer.py and add the missing libraries.

An extended discription and some shortcuts for installing Python on Windows can be found here:
    http://mientki.ruhosting.nl/data_www/raspberry/doc/windows_install_python.html

If you don't have the Python compiler already and you've Windows XP or later (most Unix systems already has Python installed), the easiest way to get Python is to install WinPython 2.7 (32 or 64 bit).
homepage:
    http://winpython.sourceforge.net/

download:
    http://sourceforge.net/projects/winpython/files/WinPython_2.7/

For unix a similar package (Python-xy) can be found here
homepage:
    https://code.google.com/p/pythonxy/

download:
    https://code.google.com/p/pythonxy/wiki/Downloads

These package contains besides the Python compiler, most of the general purpose libraries and a good IDE called Spyder.

By running the main file UHF_Spectrum_Analyzer.py (preferable from an IDE), python will tell you what libraries are missing.
For installing missing libraries under windows, you've several options (in order of complexity)

1. run from a commandline : "python -m easy_install -Z ..." for ... you substitute just the name of the package. Generally this works great because also all necessary dependies are installed
2. download the library (zip, rar, or whatever) and drag in to
3. only if you've problems installing a package in the above two ways, go to Gohlke to get a correct compiled package http://www.lfd.uci.edu/~gohlke/pythonlibs/

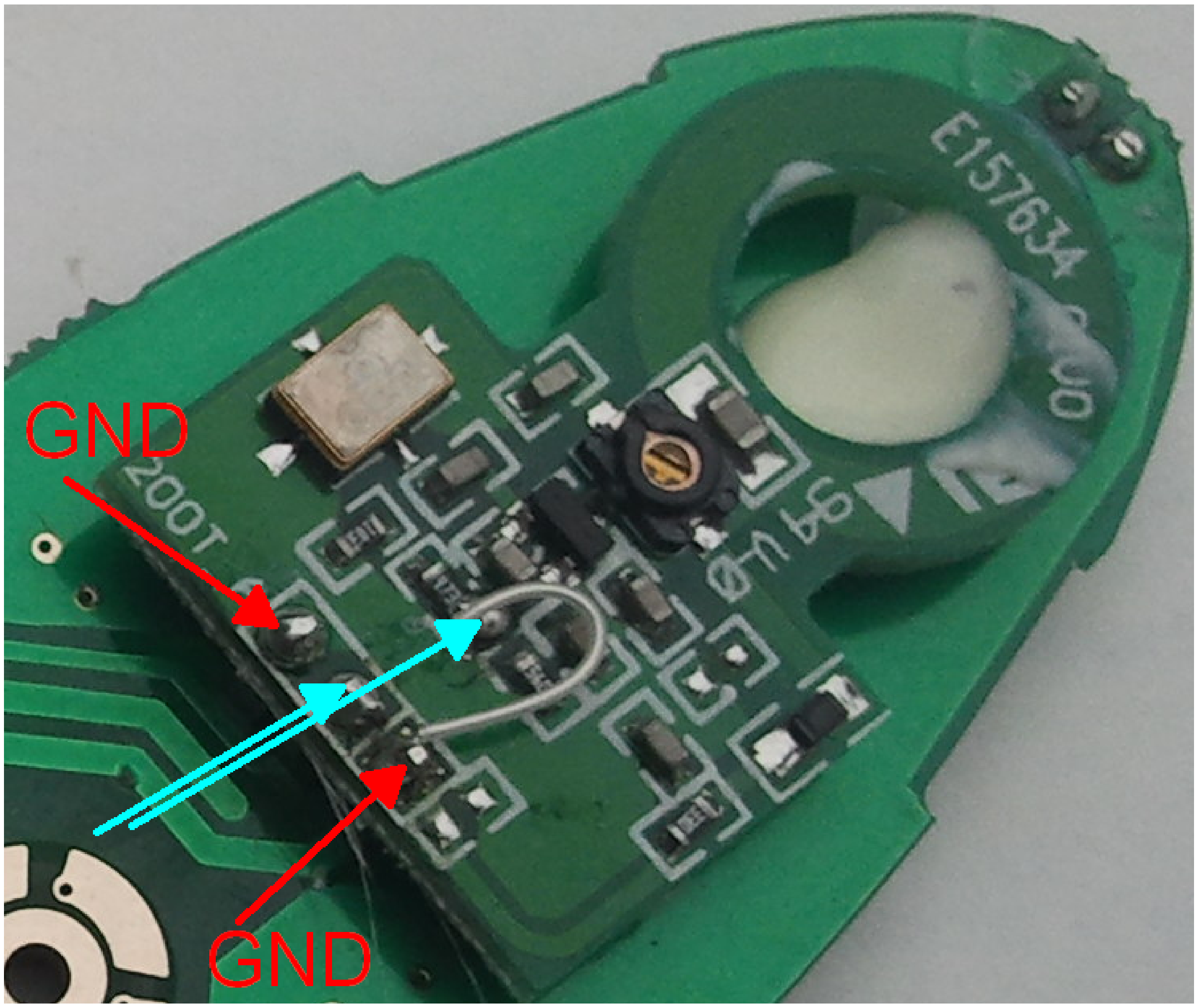
Example of message of a missing library:
```
Traceback (most recent call last):
  File "D:\_Py_Build\UHF_Spectrum_Analyzer.py", line 15, in <module>
    from   General_Globals import Application
ImportError: No module named Configobj
>>>
```
This missing library can be installed by the first method, so type in a command window:
```
python -m easy_install Configobj
```

Test frequency source

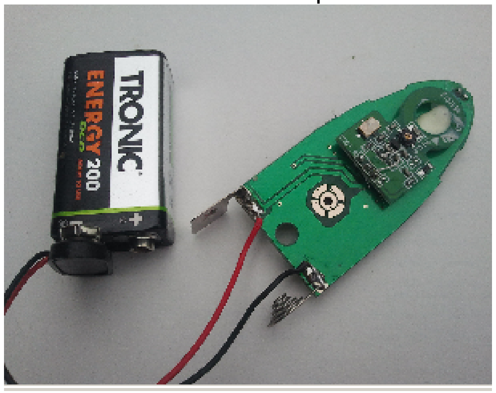
For test purposes it's handy to have a well definied frequency source. I used a cheap (10 Euro) GAO control set and modified the hand control.
Modifying the transmitter EMW200T so it'll send continuously only the carrier frequency:

- remove the wire between the 2 blue arrows
- connect the long blue arrow to the +12V

Now the module will continuously send a carrier wave, without the need for pressing a button.



The transmitter functions quiet well from a (rechargeable) 9V battery,



Evaluation of the necessary delay

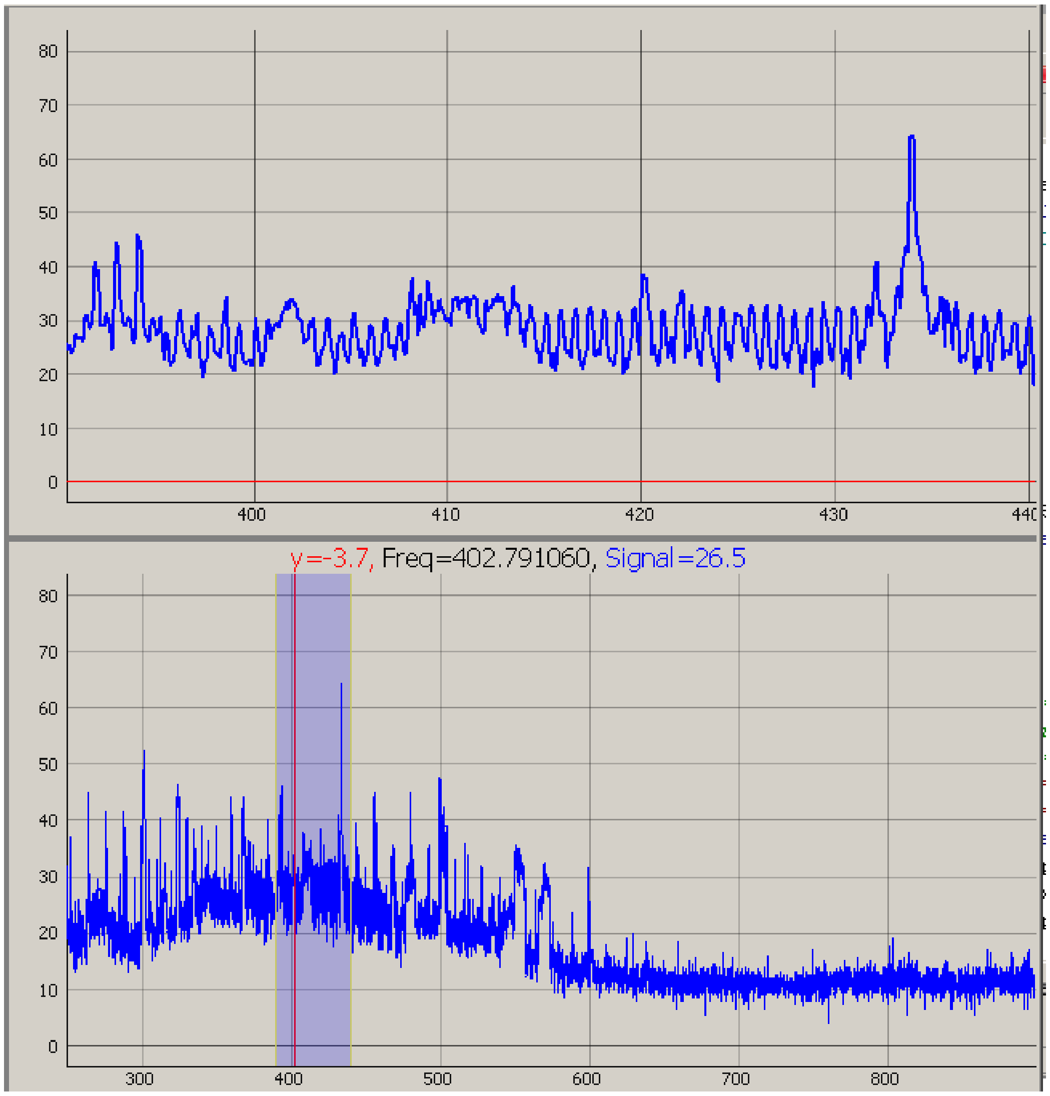
Result: 30 usec should be enough

Center Frequency = 434 MHz
Step Frequency = 20 kHz
IF-BandWidth = 200 kHz
NSamp = 10
Display Mode = Min
Average over 20 traces

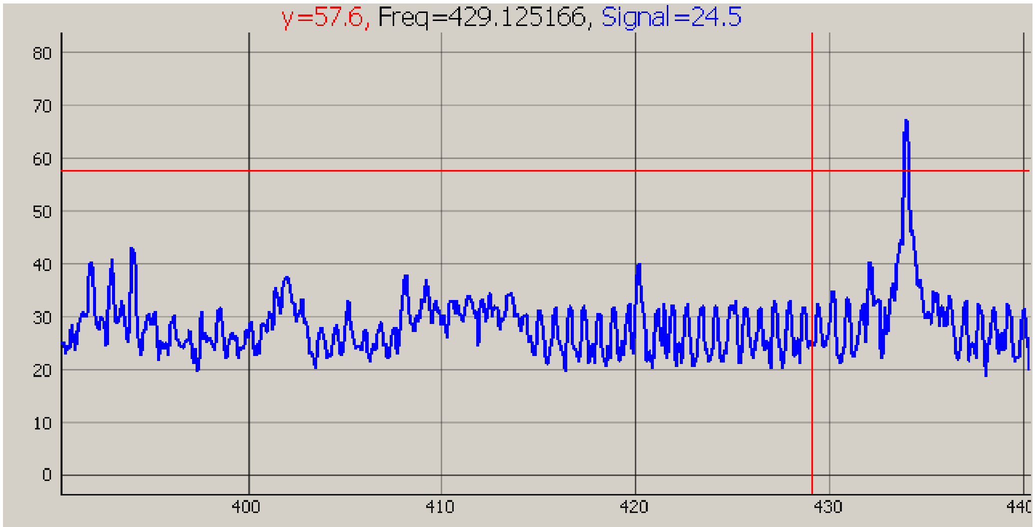
```
Code area:  5162 of 65536 used (bytes)
Data area:  766 of 3840 used
Software stack available:  3029 bytes
Hardware stack depth 3 of 31
```

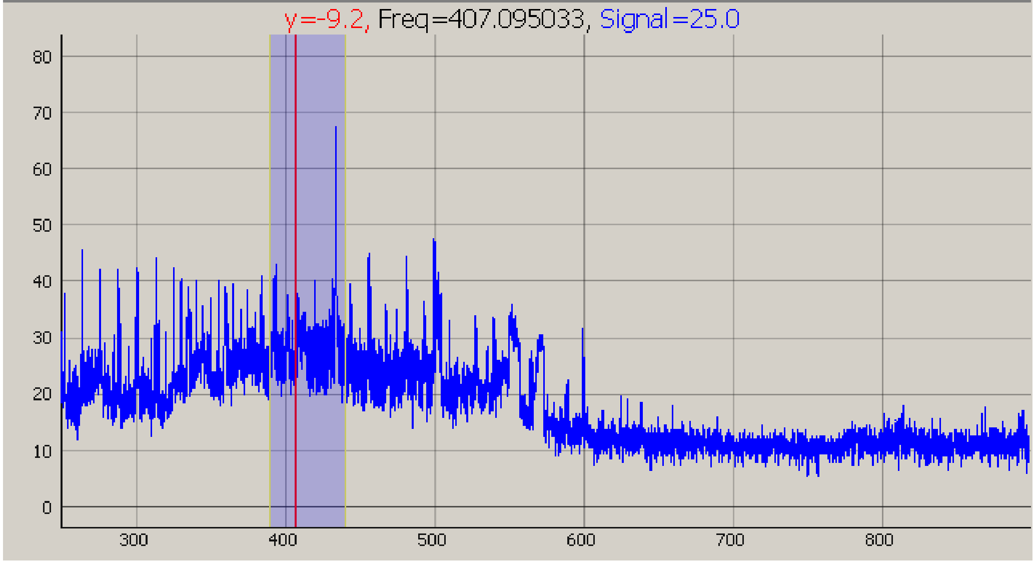
```
Code area:  5070 of 65536 used (bytes)
Data area:  511 of 3840 used
Software stack available:  3285 bytes
Hardware stack depth 3 of 31
```

full scan 30 usec

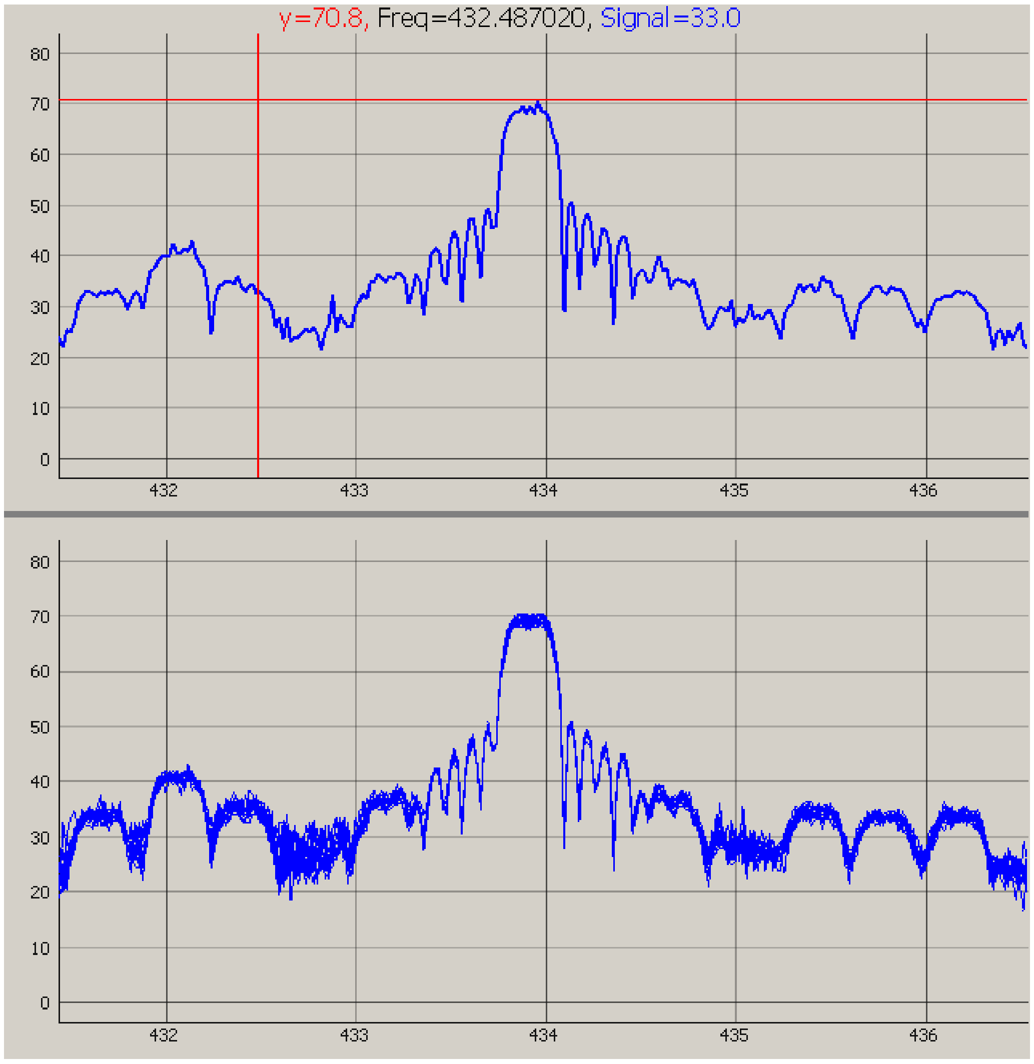


full scan 150 usec

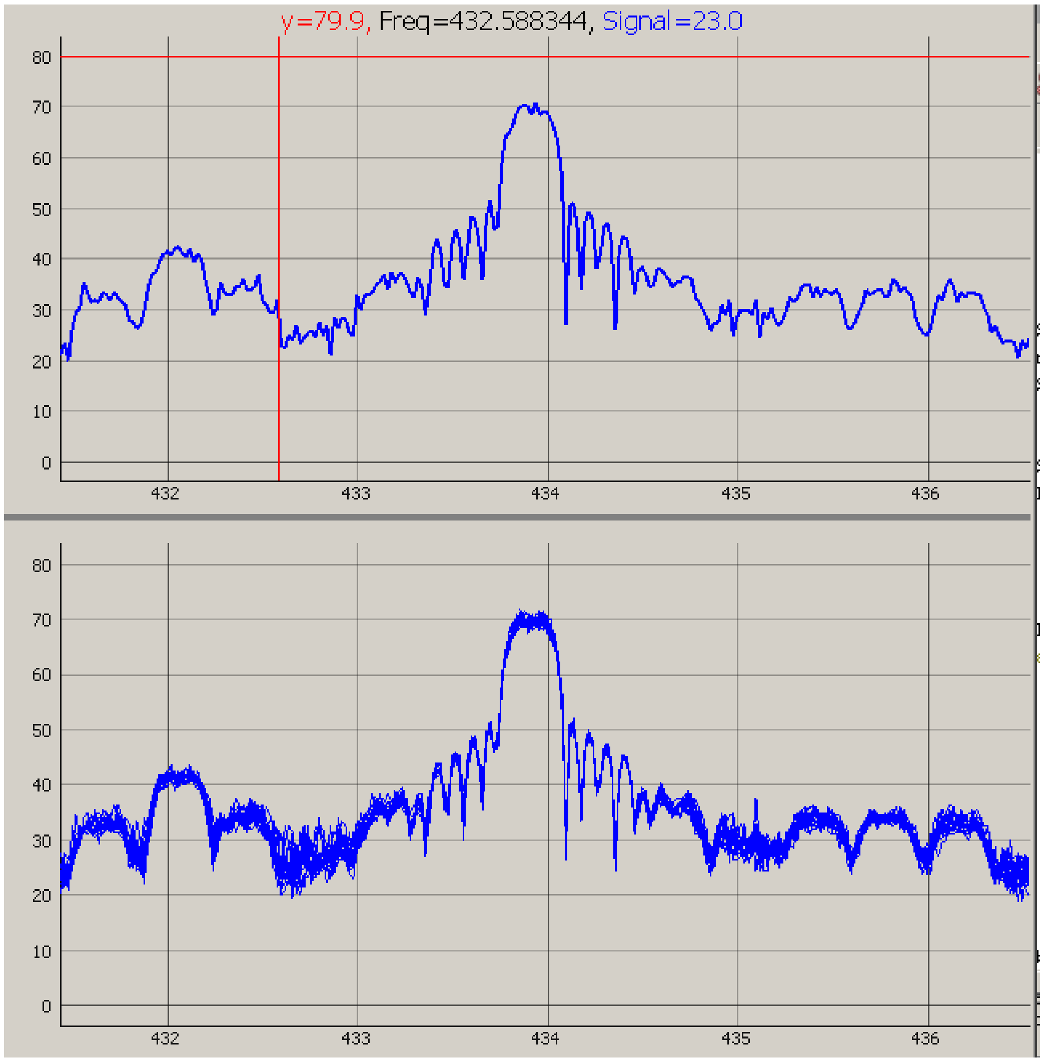




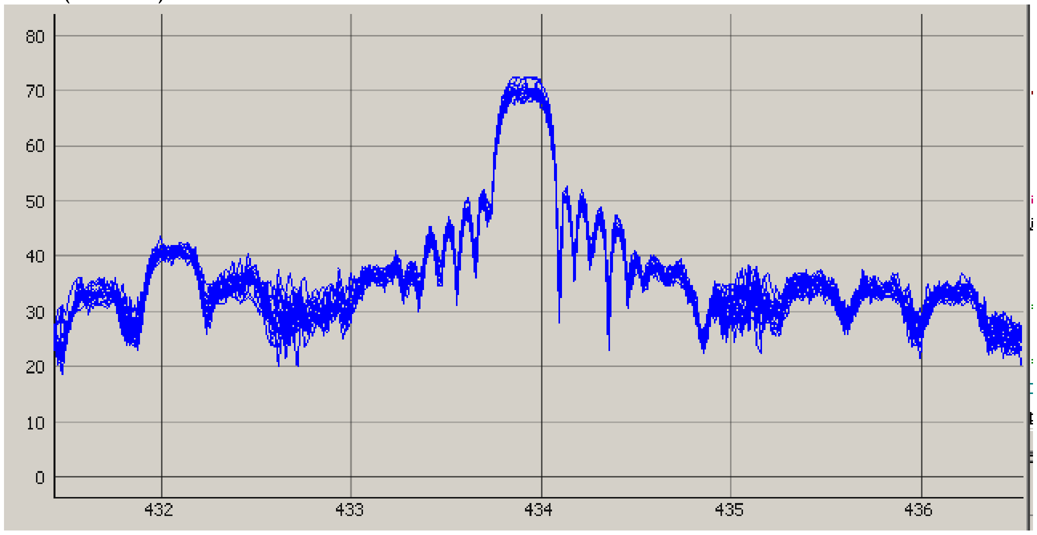
New(150usec) = 0.51 sec / trace



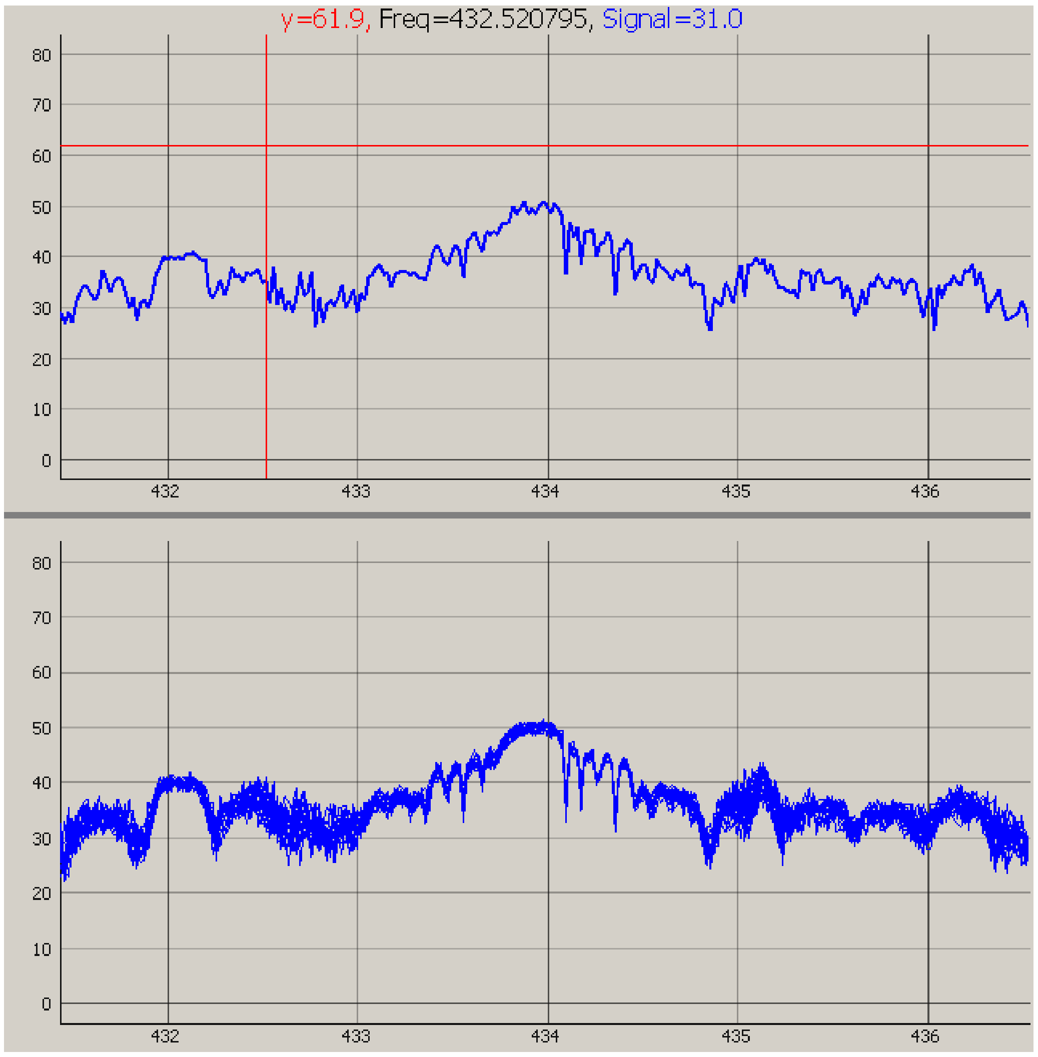
New(30usec) = 0.22 sec / trace



New (25 usec) = 0.22 sec/trace

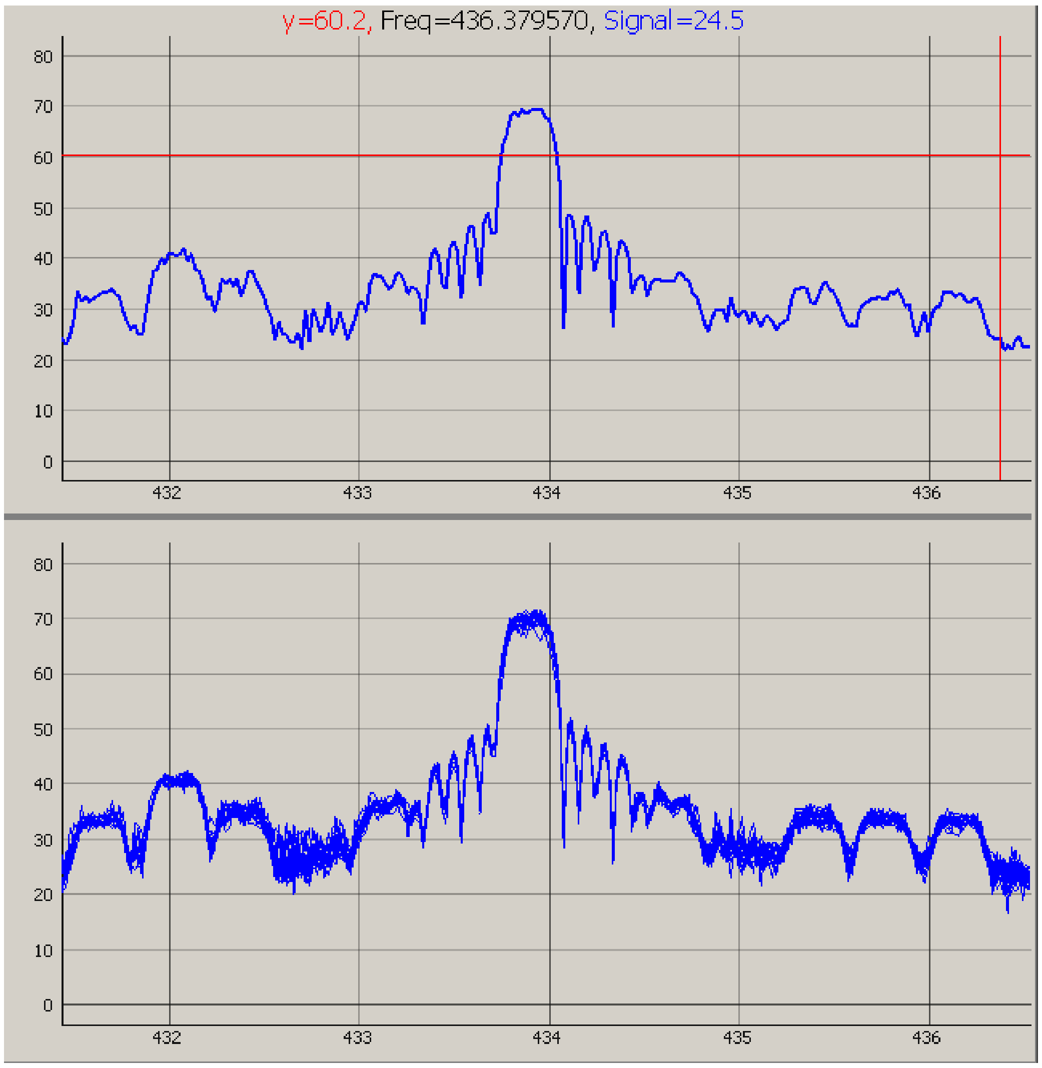


New (22 usec) = 0.21: the signal is totally distorted



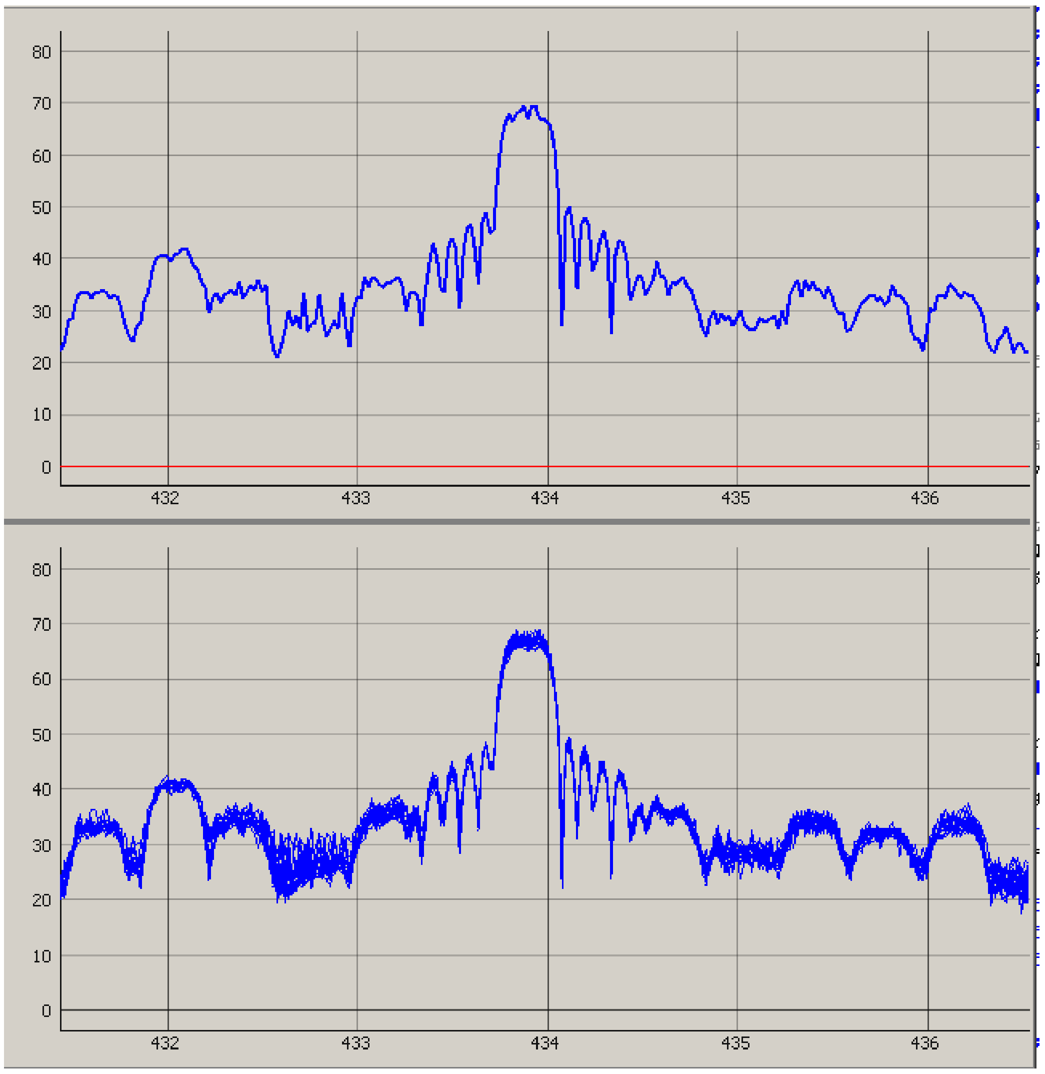
sec/trace

Old(150usec) = 0.53 sec / trace

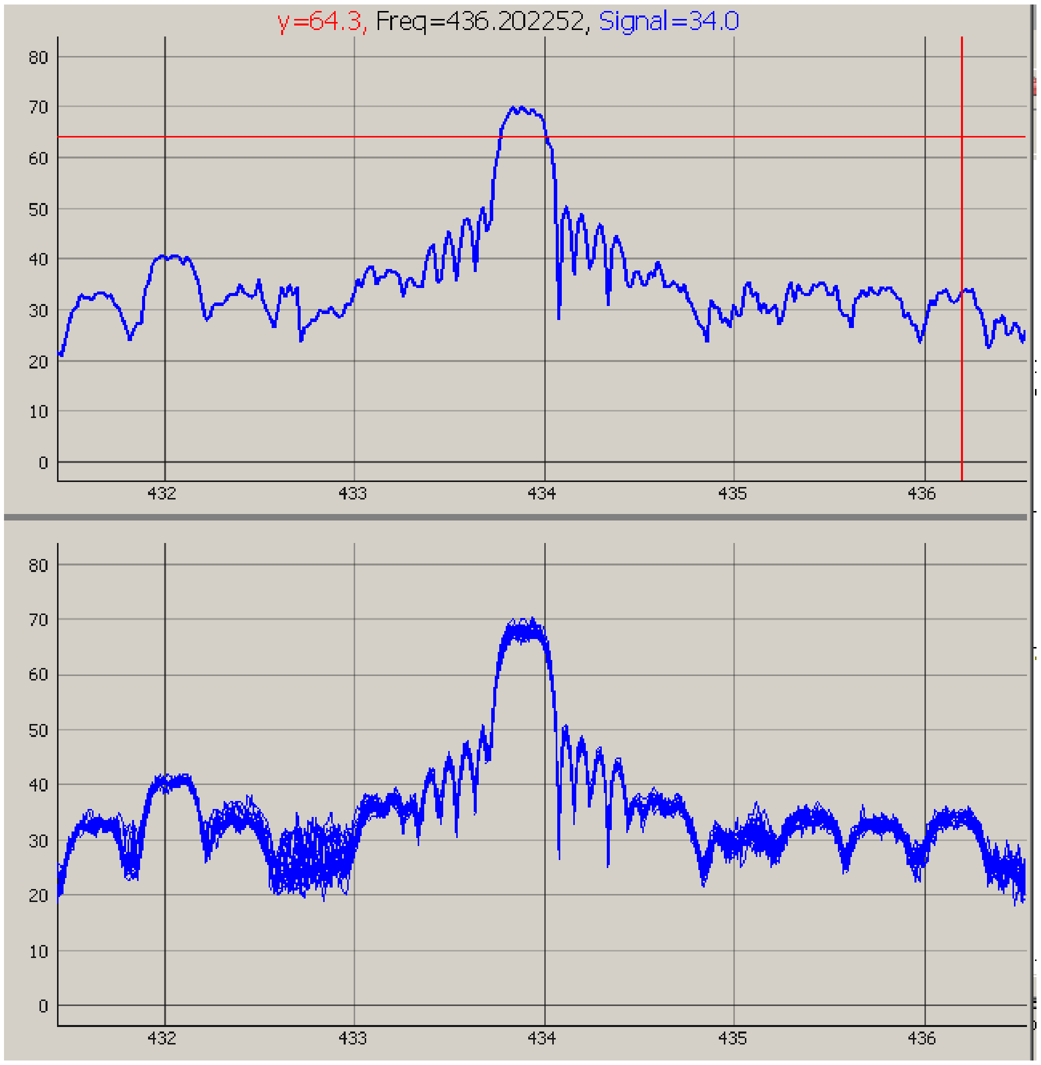


Old(50usec) = 0.30 sec / trace

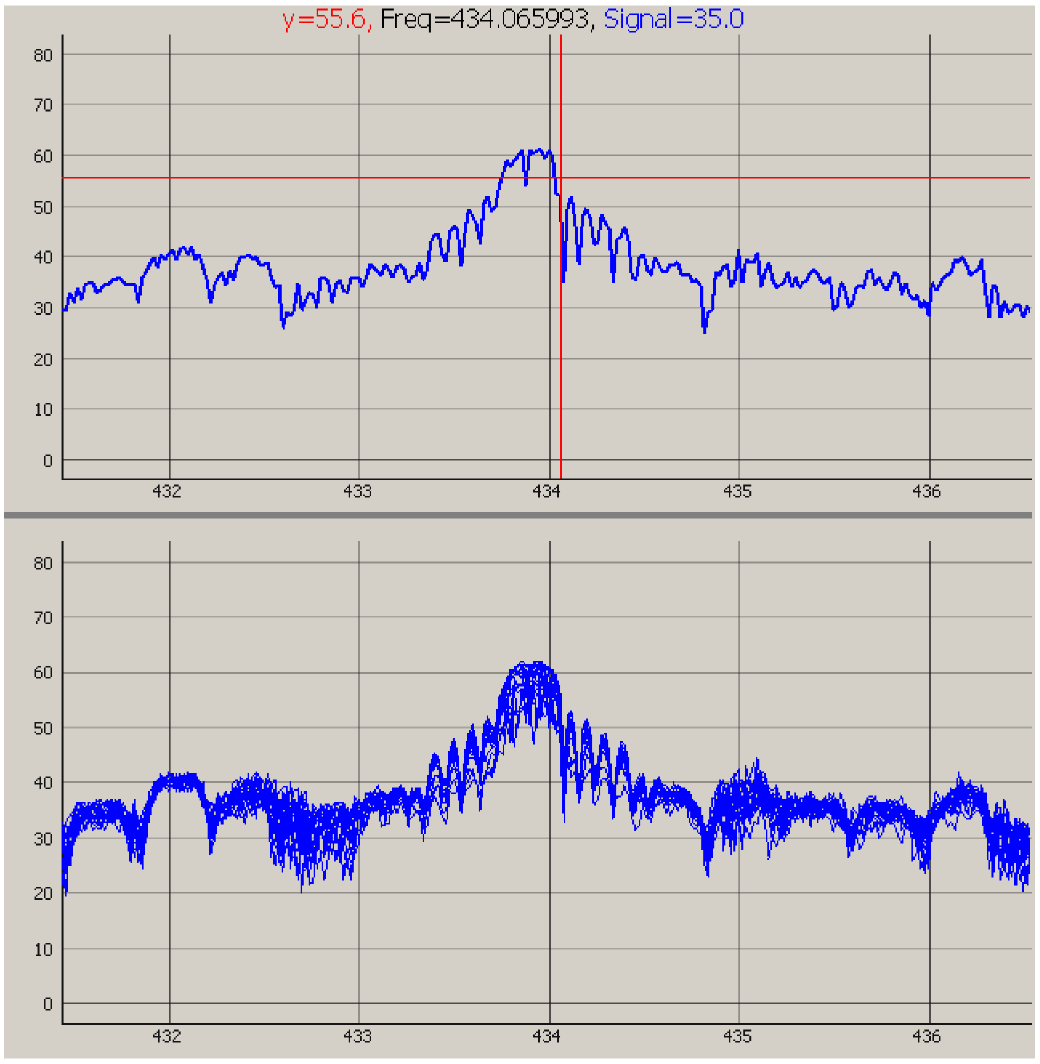
Old(30usec) = 0.24 sec / trace



Old(25usec) = 0.23 sec / trace (more variations between traces)



Old(22usec) = 0.23 sec / trace (more variations between traces)


y=55.6, Freq=434.065993, Signal=35.0
80
70
60
50
40
30
20
10
0
432
433
434
435
436
80
70
60
50
40
30
20
10
0
432
433
434
435
436