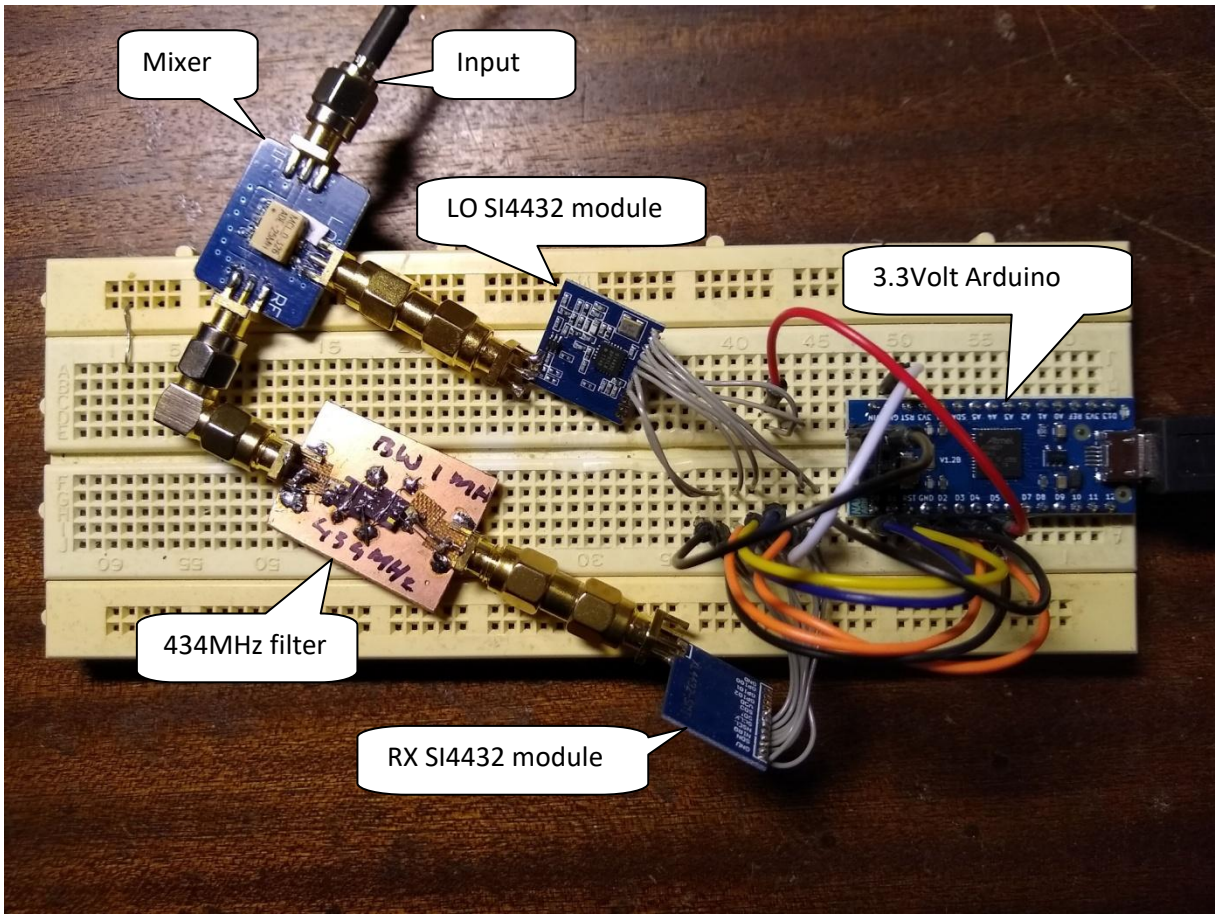
Build instructions for the Tiny Spectrum Analyzer

The Tiny Spectrum analyzer, capable to receive from zero till (far) above 100MHz, is a spectrum analyzer anybody who is able to construct something with an Arduino, can build

The main components of the minimal setup can be seen in below picture.



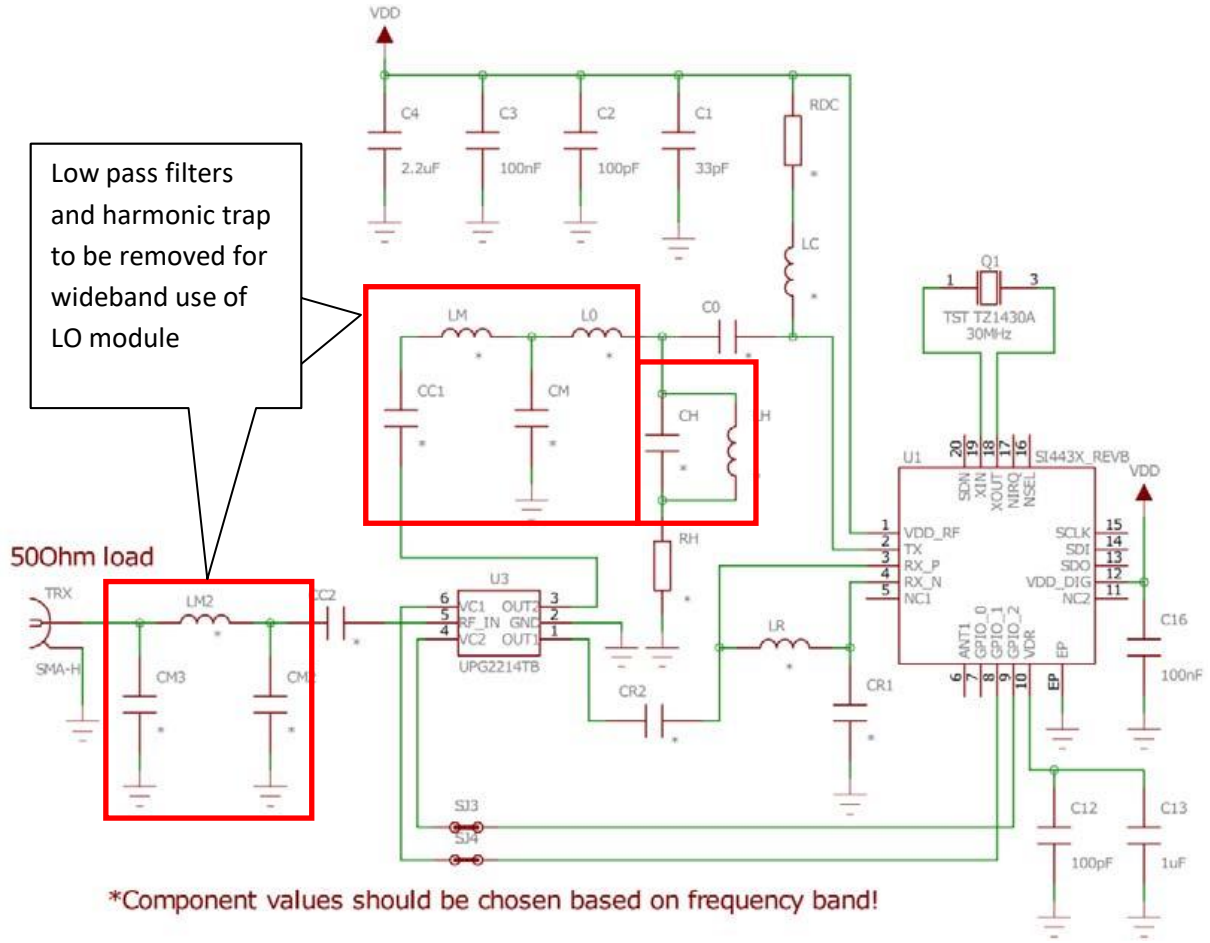
Mixer:

The mixer is preferably a level 13 or better mixer. An example of a good mixer is the ADE-25MH but any passive double balanced diode mixer (such as a ADE-1) that can be used till 500MHz on its RF and LO port is usable. The mixer should be able to handle +20dBm input on the LO port. The port connecting to the diodes (often called the IF port) is used as input for the tiny spectrum analyzer to allow the input of very low frequencies. You can buy a complete mixer module or put a mixer on a bare PCB with 3 SMA connectors.

SI4432 modules:

The SI4432 modules should be something like this: https://www.electrodragon.com/w/Si4432 and it is based on the SI reference design in the picture below. The antenna switch (small 6 pin IC) is important otherwise the LO module cannot drive the mixer with sufficient power. The schematic

contains a lot of low pass filtering. The TinySA can be used till about 150MHz without removing these low pass filters. To reach the full potential you have to, very carefully, remove these filters.



One module is used as the LO SI4432 module, the other as the RX SI4432 module

Connecting the SI4432 modules (RX and LO) to the Arduino

1	GND	Connect to ground of the Arduino
2	GPIO0	Do not connect
3	GPIO1	Do not connect
4	GPIO2	Can be used as a fixed frequency output, after reset set to 1MHz
5	VCC	Connect to 3.3V of the Arduino. An additional 100uF capacitor from 3.3V to ground (not shown in picture) may be required for stable operation.
6	SDO	Connect to D3 of the Arduino (WARNING: You MUST use a 3.3V compatible Arduino)
7	SDI	Connect to D2 of the Arduino (WARNING: You MUST use a 3.3V compatible Arduino)
8	SCLK	Connect to D1 of the Arduino (WARNING: You MUST use a 3.3V compatible Arduino)
9	nSEL	Connect the RX module to D0 and the LO module to D5 of the Arduino (WARNING: You MUST use a 3.3V compatible Arduino)
10	nIRQ	Do not connect
11	SDN	Connect to ground of the Arduino
12	GND	Connect to ground of the Arduino
13	Antenna	Connect Antenna from LO to LO port of mixer and Antenna of RX to the output of the 434MHz filter
14	GND	Connect GND from LO to GND of mixer and GND of receiver to the GND of the 434MHz filter

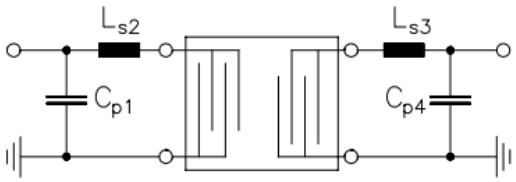
Connect the RF port of the mixer to the input of the 434MHz filter.

The LO SI4432 module has a low pass filter in the TX output of the SI4432. Removing this filter enables usage above 150MHz.

434MHz Filter:

The 434MHZ band pass filter should preferably have a bandwidth of 1MHz. A filter with a bandwidth of 4MHz or wider may lead to spurs.

The home build 434MHz filter uses this 50 ohm matching network

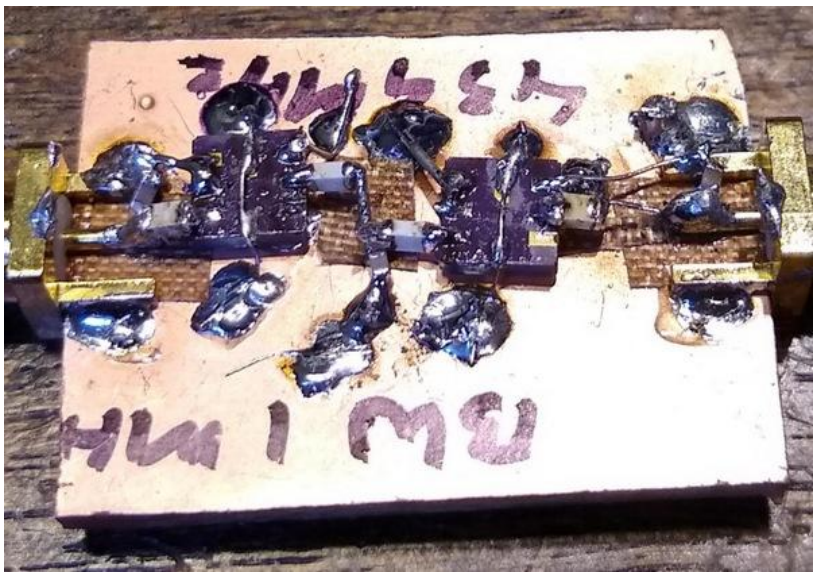


C_{p1} = 3.3 pF
L_{s2} = 33 nH
L_{s3} = 33 nH
C_{p4} = 3.3 pF

but instead of one filter, two are used for better image suppression.

Make sure you check the datasheet on what pins to use as input/output. For the SAW filters I had available I made a dead bug style ugly construction as can be seen in below picture

Important is to connect all ground pins with separate wires to the ground. The matching capacitors are not that important and depending on the chosen SAW filter you may also leave out the inductors.



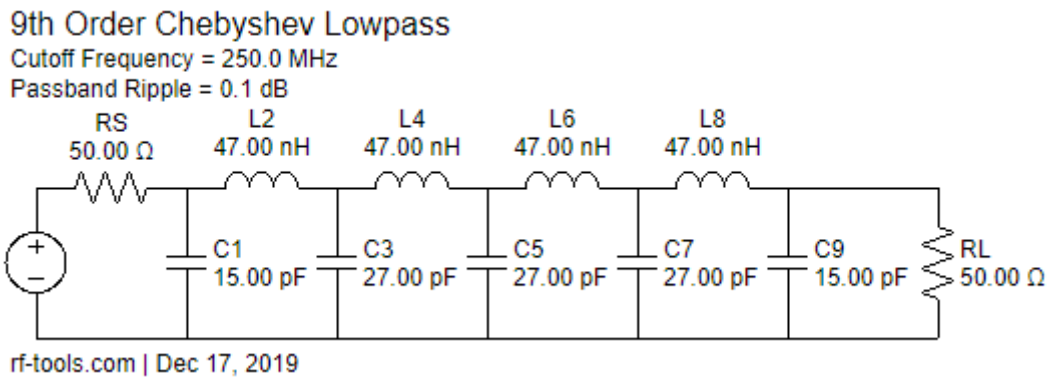
I'm using Teflon tape to isolate the center SMA pins from the PCB copper layer and that works very well but a small piece of paper also works.

Be aware the center pins can rotate and if they rotate you will destroy the matching capacitors. Be careful when connecting the filter to the other modules.

Optional component: Low pass filter

The IF port of the mixer is the input of the tiny spectrum analyzer. A low pass filter between the input of the SA and the mixer IF port will help in reducing aliases. Not having a low pass filter allows usage of harmonic modes but be aware of aliases (frequencies above 434MHZ folding back to below 434MHz)

For low-pass filter at the input you could use this design using standard E12 values



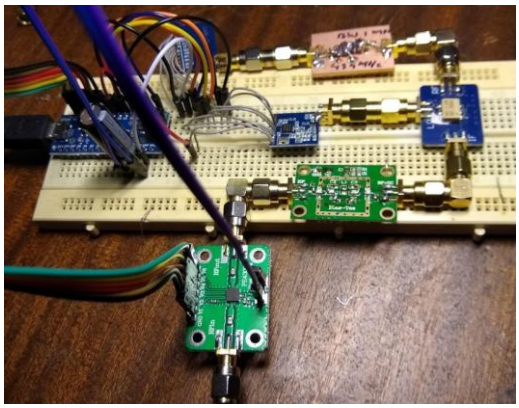
that can easily be build on a old bias-tee PCB or a bare PCB resulting in

With this low-pass filter most of the aliases resulting from signals above 433MHz will be gone.

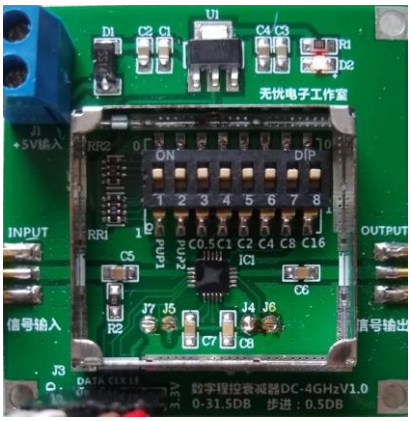
Optional component: Input attenuator

For some measurements an input attenuator between the input and the low pass filter is essential. Adding a step attenuator is pure convenience as manually adding input attenuation works also.
I went for a readymade module found on eBay that uses the affordable 3.3Volt compatible PE4302 step attenuator.
There are two versions, one using parallel (6 parallel bits) input and the other with serial (e.g. a data and clock line) input to control the 64 attenuation steps from 0dB till -31.5dB
There where sufficient free bits on the Arduino zero so I choose parallel as can be seen in this picture of all modules together. The step attenuator is at the middle bottom, before the low pass filter.

Attenuator Module connection	Arduino connection
GND	GND
Vcc	3.3V
V1	D6
V2	D7
V3	D8
V4	D9
V5	D10
V6	D11



It is also possible to use a serial PE4302 module like this:



Serial mode is selected with J7 and J4 closed and J5 and J6 open

Serial Attenuator Module connection	Arduino connection
GND	GND
3.3V	3.3V
LE	D10
DATA	D2
CLK	D1

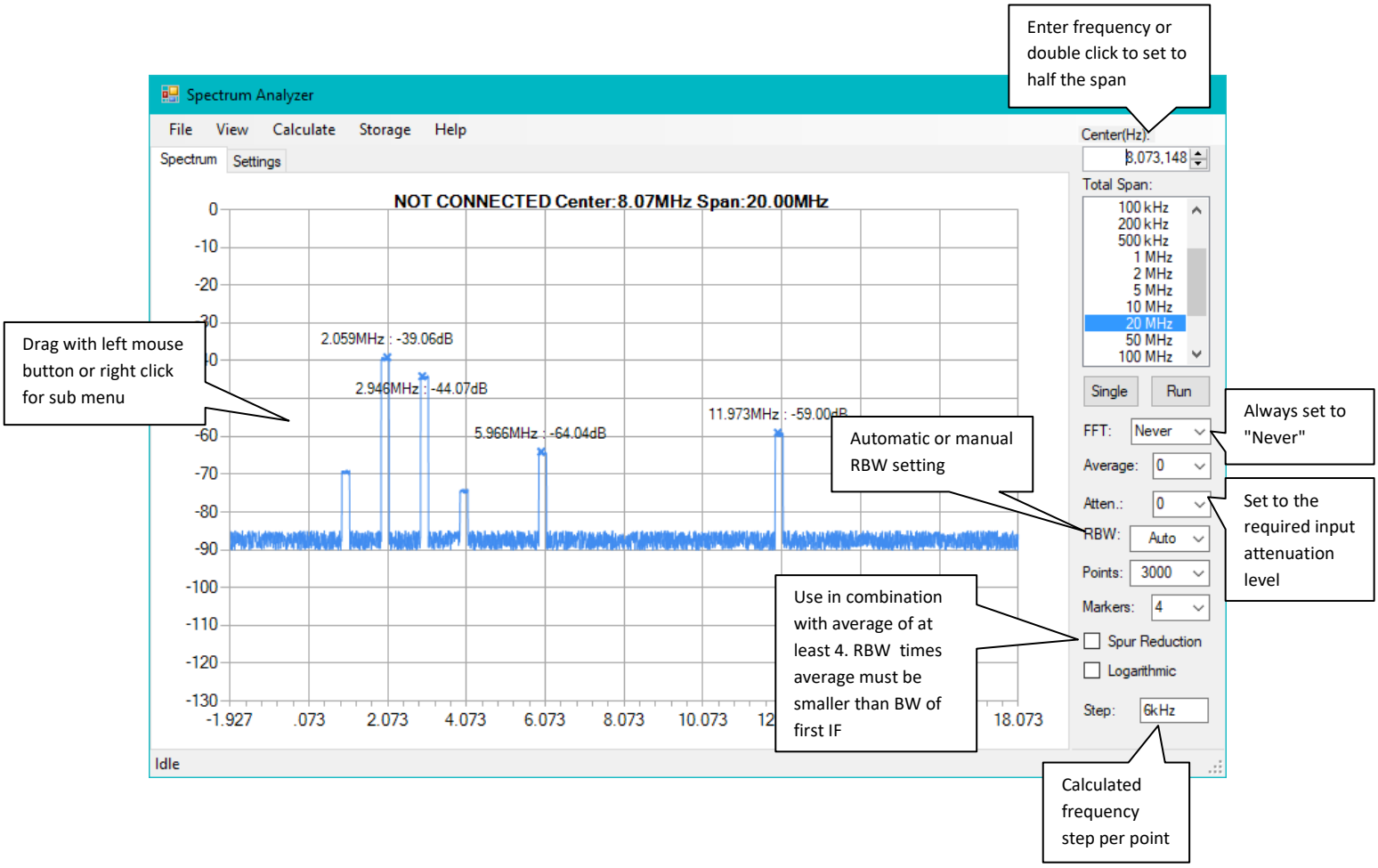
Control SW.

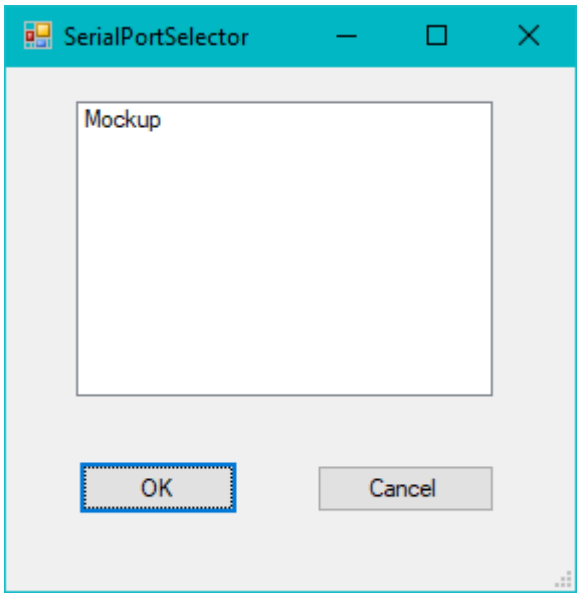
A small Arduino program to control the SI4432 and the attenuator is available at https://groups.io/g/HBTE/files/Tiny%20Spectrum%20Analyzer

There you will also find the windows program to control the Tiny SA. No installation is needed. You may have to install VC++2010 Runtime 32 bit from here:

http://www.microsoft.com/en-US/download/details.aspx?id=8328

Using the SA.exe program:





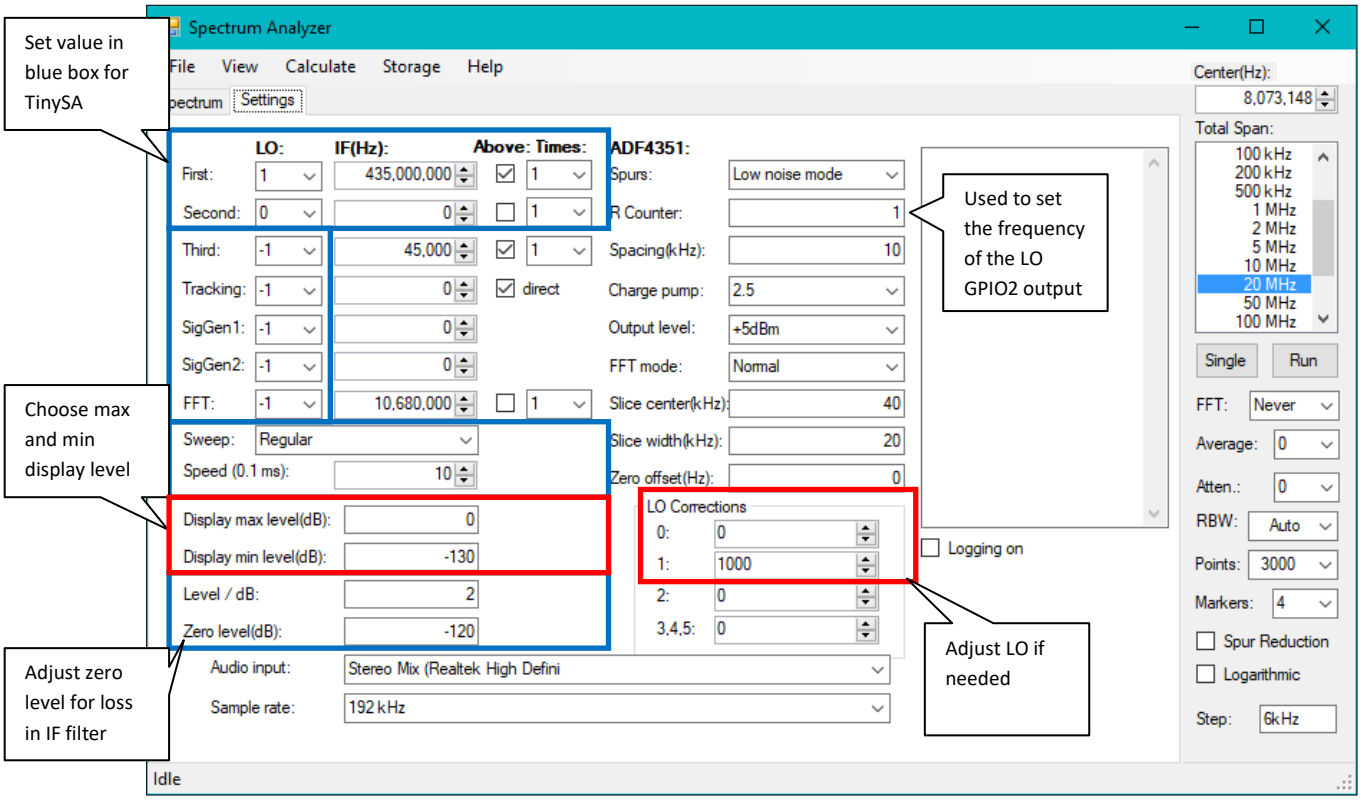
This dialog appears when you click "File"/"Connect"

Select "Mockup" and click OK to run SA.exe in simulation mode.

Any COM port with a connected TinySA will be shown.

If no COM ports listed (like shown at the left) you can only use the mockup device

Settings tab:



Protocol between SA.exe and Arduino

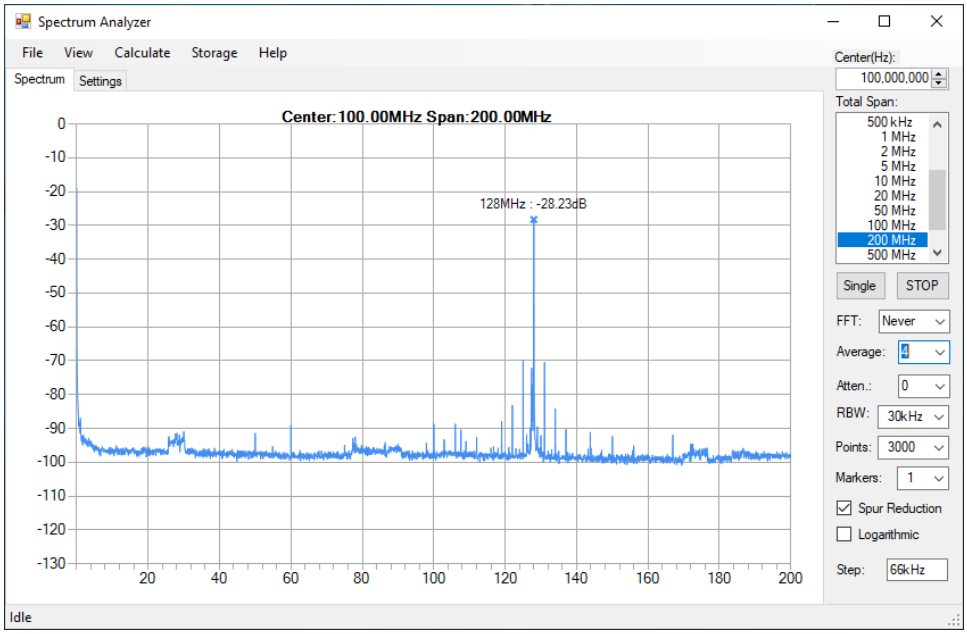
A 100	Set start frequency of scan to 100Hz
B 200	Set end frequency of scan to 200Hz
S 20	Set amount of steps to 20
T 30	Set time per step to 30 * 0.1ms
M	Make one sweep with selected SI4432
P5 23	Set attenuator to -23dB
P6 0	Use SI4432 #0 for signal strength input
X68 45	Write 0x45 to register 0x68 of selected SI4432
X68	Read register 0x68 of selected SI4432
V1	Select SI4432 #1
O12345	Set selected SI4432 to 12345Hz
W100	Set bandwidth of selected SI4432 to 100kHz
P1 2	Set the GPIO2 output of the LO SI4432 to 10MHz

When doing a sweep the Arduino replies with a string of bytes . Format: '{', SSI,SSI,...,'}'. Each SSI value is two 8 bit bytes

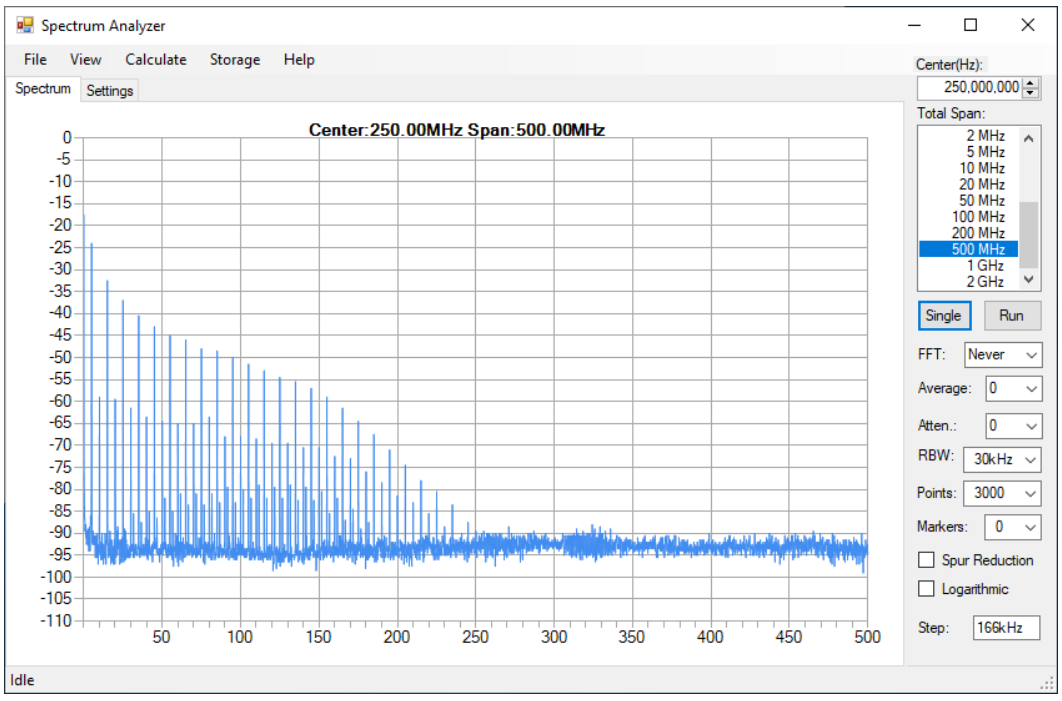
Setting P6 to 2 put's both SI4432 in TX mode enabling the signal generator application.

Measurement examples:

This first measurement example is measuring the output of a SI5331 at 128MHz through a 30dB attenuator to check the spurs of the fractional output divider. Notice 4 times averaging is used in combination of spur reduction to get an nice clean picture. The bumps at 30MHz, 80MHz and 17MHz are aliases from a cell phone base station nearby

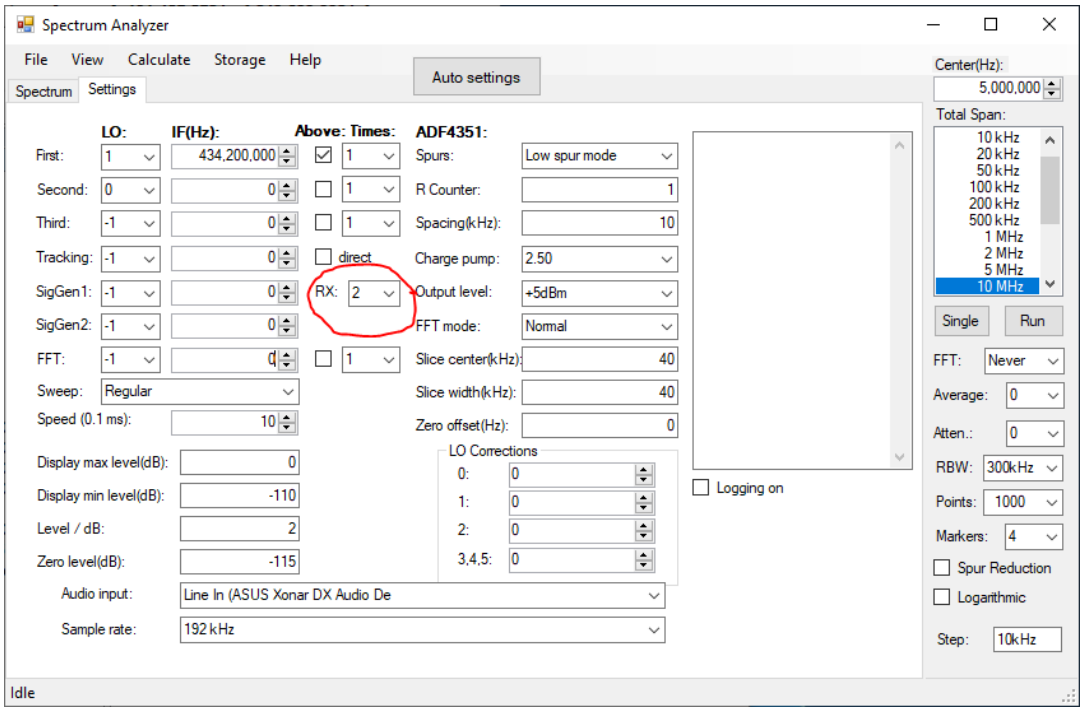


The second example is the same SI5331 set at 5MHz. The harmonics of the square wave are nicely visible but nothing above 250MHz. It serves to illustrate what happens when you do not remove the low pass filter on the LO SI4432 module as the sensitivity of the tinySA quickly goes down above 150MHz. But for most HF measurements this is not a problem.



Usage as zero till 250MHz signal generator:

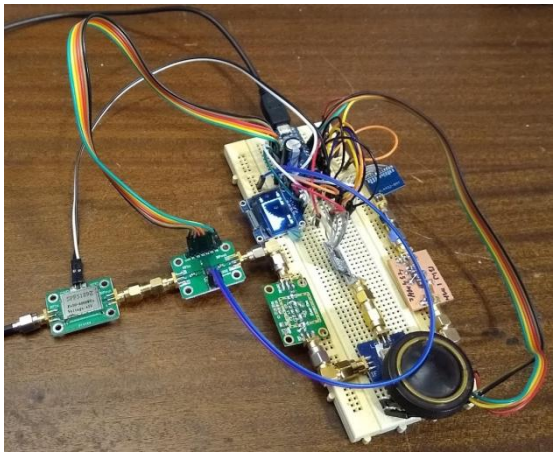
As the AD9851 modules are becoming scarce and much more expensive it would be nice if the TinySA could be used as a signal generator. The whole signal path from input to the RX SI4432 is bidirectional so I decided to test. In the SA.exe I added the ability to put both SI4432 in TX mode (see setting in red circle)



So without changing the HW configuration the input of the TinySA becomes the output of a signal generator able to output 0-250MHz of a fairly acceptable sinus where you can control the frequency by setting the center frequency in SA.exe (above set to 5MHz) and the output level (maximum -2dBm, minimum -32dBm) by setting the attenuation.

Adding local control and a display:

Adding a cheap I2C controlled display and a rotary control is simple. Make some connections and let the Arduino SW read the rotary control and steer the display.



The display I used is 128 by 64 OLED pixels. With the rotary control you can set From(MHz), To(MHz), Max display(dBm), Min display(dBm) and Attenuator(dB). The resolution bandwidth is set automatic.

Two pictures to demonstrate the result. First a 150MHz wide scan to see if there is any signal. Vertical scale is set to -100dBm till -20 dBm

Next zooming into a 5MHz till 10MHz range

The reduction in noise level due to the reduced resolution bandwidth is clearly visible.

The latest tinySA.ino SW includes an automatic peak search function and display the frequency and level of the largest signal on the display.

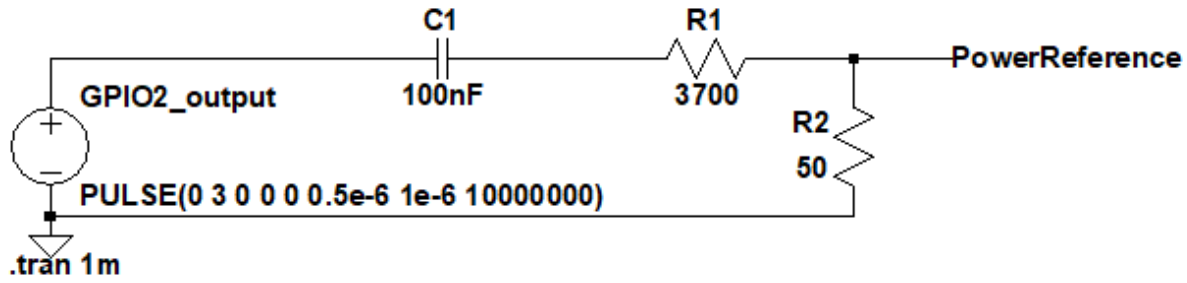
Calibrating the power level.

Using the GPIO2 output it is possible to create a fairly accurate -30dBm power reference level as the output swing of the GPIO port is 3 Volt when the current is below the drive limit.

The frequencies can be set according to below table. The output frequency is controlled through the "R counter" field in the settings panel of SA.exe. When instead using the Arduino serial interface you can select the LO SI4432 using the V1 command and then set output frequency using the P1 parameter (example: "P1 2" to set the output to 10MHz) .

Setting of "R counter" field in SA.exe	GPIO2 output frequency
0	30 MHz
1	15 MHz
2	10 MHz
3	4 MHz
4	3 MHz
5	2 MHz
6	1 MHz

Using a DC blocking capacitor (C1) and a voltage divider (R1/R2) provides a 50 ohm matched output and reduces the load on the GPIO port below the current limit.



The PowerReference output provides a -30dBm power level at the fundamental of the selected GPIO2 output frequency

If the 3k7 resistor cannot be found a 3k9 resistor can be used. This will provide a -30.4 dBm output power

For best accuracy during power calibration you should use a -10dB or more attenuator setting to ensure the input impedance of the tinySA is close to 50 ohm. This because the input impedance of the mixer is rather ill defined.