7th ANNUAL TechCon 2021
WCF Country
West Central Florida Section

Gnuradio

February 27, 2021

Tom McDermott, N5EG

Medford, OR

Life Member ARRL

Life Senior Member IEEE

Outline

- What is Gnuradio?
  - Where do I get it?
  - Are there tutorials?
- Core Concepts.
- Using gnuradio:
  - Simulation, learning the fundamentals of DSP.
  - Test equipment.
    - Example: measuring receiver Noise Figure
  - A radio (of course).

What is Gnuradio?

- A *free* Open Source software package. GUI and Command line.
  - Linux, Windows, Mac.
- Create and run DSP without writing code.
  - Great way to learn signal processing.
- Includes virtual instrumentation
  - Oscilloscope, Spectrum analyzer, Waterfall, Constellation plotter.
  - Signal *sources* and *sinks*. Radios, files, sockets, pipes.
  - User-created add-ons (gui widgets, specialized functions).
- Supports a long list of radios (Ettus, OpenHPSDR / Red Pitaya, Pluto, plus many others).
- Based on Python and C++
  - Python for the interconnection, graphics, management.
  - C++ for high-speed DSP functions, buffers, and I/O.

Getting Started with Gnuradio

(thanks to John Petrich, W7FU)

How to install GNU Radio

https://wiki.gnuradio.org/index.php/InstallingGR

Guided Tutorials (Novice level)

https://www.youtube.com/watch?v=N9SLAnGlGQs&list=PL618122BD66C8B3C4

Tutorials (More advanced Level)

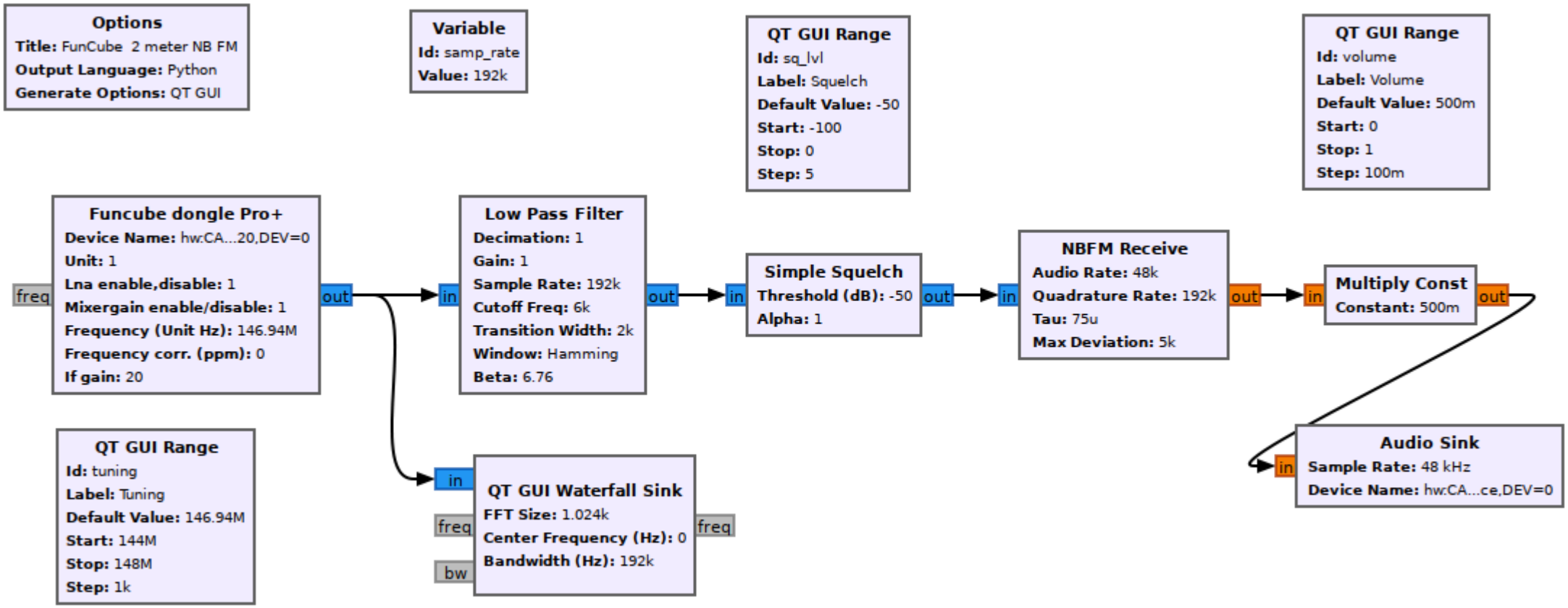
https://wiki.gnuradio.org/index.php/Tutorials

Gnuradio flowgraph

- You don’t need to know Python or C++
  - Gnuradio Companion (GRC) is the graphical GUI.
    - Allows you to create flowgraph modules, parameters, and wiring using a GUI.
    - Compiles your flowgraph into Python program and saves the file to disk.
    - Allows you to Start and Stop your flowgraph.
- A flowgraph is a Python program:
  - Automatically generated by GRC.
  - A text file containing the Python source code:
    - Defines all the DSP blocks, GUI blocks, Radios, Sources, Sinks.
    - Defines all parameter values, how everything is wired up.
    - Defines what your flowgraph looks like on the GUI display.

Flowgraph Example

2 meter NBFM receiver (from the gnuradio wiki) using Funcube dongle



Gnuradio Core Concepts

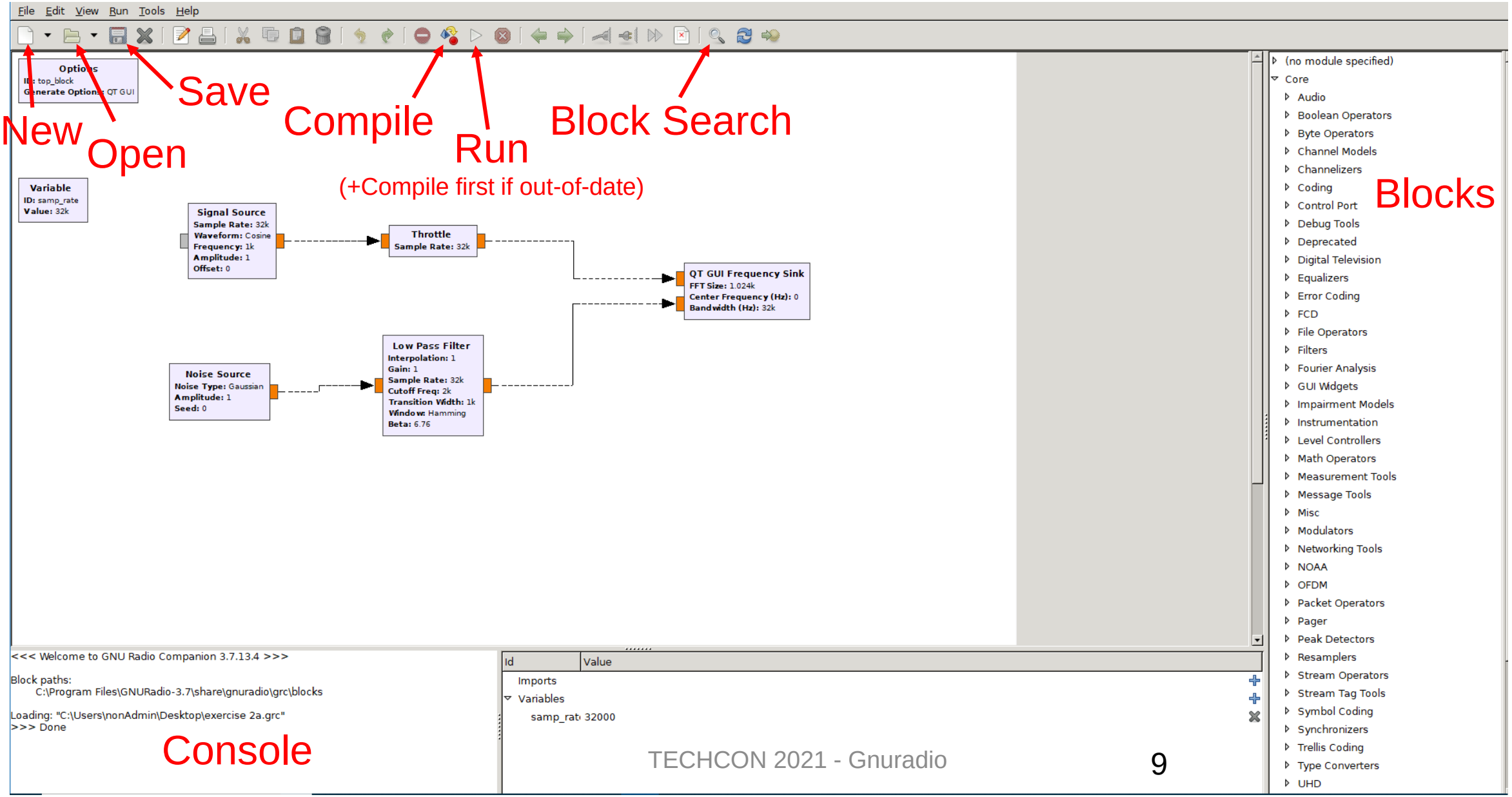
- A '*source*' is something that streams samples into your flowgraph.
  - A radio, audio soundcard, file, TCP connection, etc.
- A '*sink*' is something that removes samples from your flowgraph.
  - A radio, audio soundcard, file, TCP connection, etc.
- In general a block receives samples, transforms them, might do other things, then outputs modified (or not) samples.
  - Examples of general blocks:
    - Low Pass Filter
    - FM Demodulator
    - Delay block

Gnuradio Core Concepts - 2

- Gnuradio handles real-time buffering.
- There cannot be a loop in a flowgraph.
- A flowgraph can be:
  - Simulation only, or include radios, files, soundcards, sockets, pipes.
- A ‘throttle’ is needed *if* there is no source of physical timing.
  - A radio, or a soundcard (output or input) are physical sources of timing.

- More details in the Appendix.

The GRC Main Window

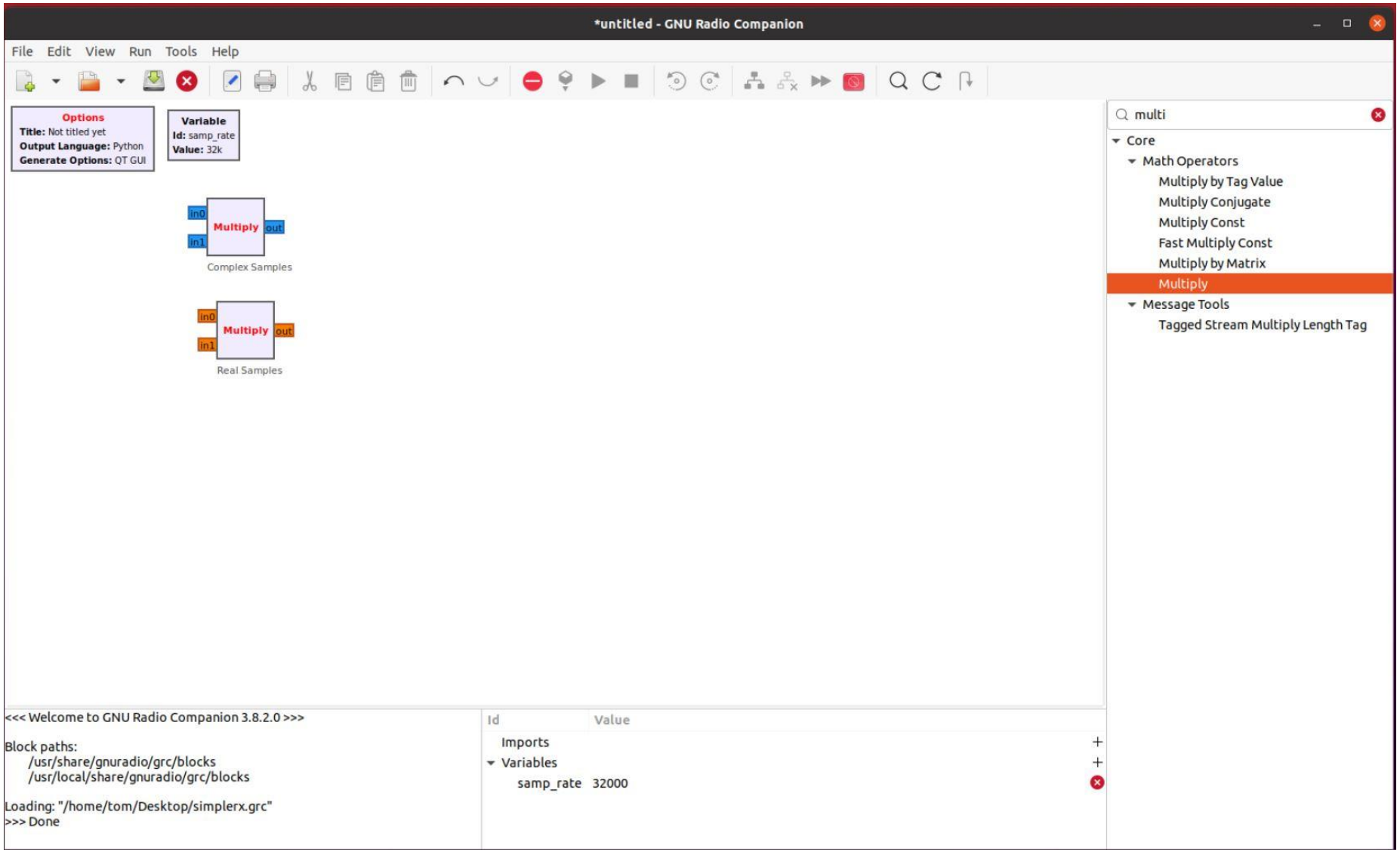


- 1st Demo – simulation of signals and instrumentation.
  - Using the GUI to:
    - Create a new flowgraph
    - Wire up instruments
    - Run the simulation
  - Using ‘real’ datatype.

Gnuradio Core Concepts - 3

- Signal formats are color-coded.
  - Blue: Complex single-precision float 32 (I + Q).
  - Orange: Single-precision float 32.
  - A bunch of others. Help→Types to display handy pop-up legend.
- GUI will only wire together two pins if they are of the same type.
- Select block then use arrow-up and arrow-down to scroll through the types supported by the block.
  - Alternatively double-click to block to open, modify the types.
- There are type-converter blocks available.
  - Example: Complex → Float has one blue input and two orange outputs.

Color-coded signals in the GUI



Complex vs. Real

- A real signal contains mirrored positive and negative frequency components.
- A complex signal (I+Q) allows differentiating the positive from the negative signals.
  - Complex negative and positive signals can be different.
  - Possible to eliminate the negative frequency component (or vice-versa).
- Almost all radio processing uses complex signals.

- More details in the Appendix

Multiplier (a mixer)

- Multiply using *real* numbers produces sum and difference frequencies.
- Multiply using *complex* numbers *does not* produce sum and difference frequencies – only produces the sum frequency.
- A complex multiply can produce a frequency shifter.
- Negative-frequency carrier used to down-convert (shift down in frequency). Two ways:
  - Enter frequency as a negative number.

    or
  - Take complex conjugate of positive frequency (negates the imaginary part).

- 2nd Demo – simulation of complex signals.
  - Using ‘complex’ datatype.
  - Using a multiplier as a mixer.
  - ‘real’ mixers vs. ‘complex’ mixers.

Gnuradio Core Concepts - 4

- Each block must know the sample rate of the samples coming into it.
- Some blocks can *change* the sample rate.
  - Decimation': LPF then 'Keep one-in-N'
  - Interpolation: insert N-zeros between each sample, then LPF.
  - Nyquist Bandwidth: Not only a good idea, it's the law!
- Gnuradio creates one default variable samp_rate.
  - Interpolation or decimation changes the sample rate downstream.
    - Probably want additional sample rate variables.
  - Some gnuradio blocks combine a function with decimation.
    - For example Low-Pass-Filter has adjustable decimation.
- Tip: Reduce sample rate close to the source to decrease CPU workload.
- More details in the Appendix

Some Key Gnuradio Modules - 1

- Sources
  - Sine wave, noise, constant, null, etc.
- Add
  - Adds two signals sample-by-sample.
- Multiply
  - Multiply two signals sample-by-sample.
  - Implements mixer (frequency shifter), gain / attenuate.
- Converters
  - Float → Complex
  - Complex → Float
  - Stream → Vector
- GUI : scope, spectrum analyzer, constellation, waterfall.
- Many more. Explore on your own.

Some Key Gnuradio Modules - 2

- Filters. General types: Lowpass, Bandpass, Highpass.
  - Many easy-to-use pre-made filter types available.

- Can make custom filters if necessary (advanced topic).
  - Custom filter transfer function defined by the taps.
  - GRC includes a filter designer GUI.
- More details in the Appendix

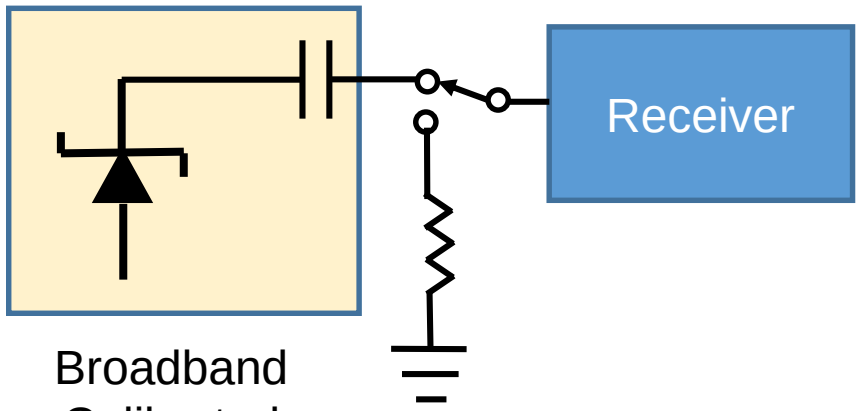
Test Equipment Example

- Virtual instrumentation in gnuradio can provide useful test equipment.

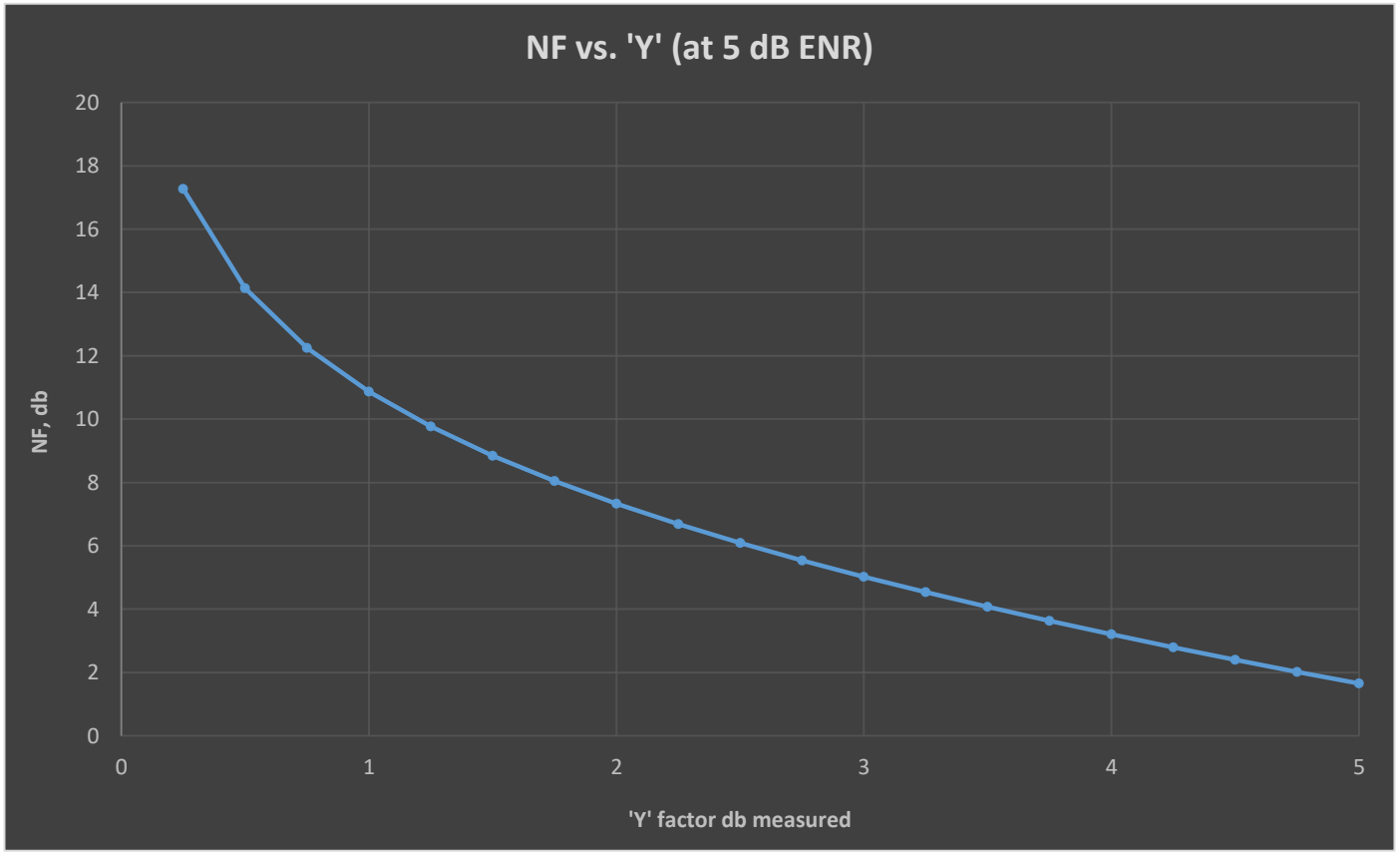
- Measure the Noise Figure (NF) of a receiver
  - Use Y-factor method.
    - ‘Y’ is the increase in noise (db) when the noise source is turned on.

$$NF = 10 * \log_{10} (10^{(ENR/10)} / (10^{(Y/10)}-1))$$

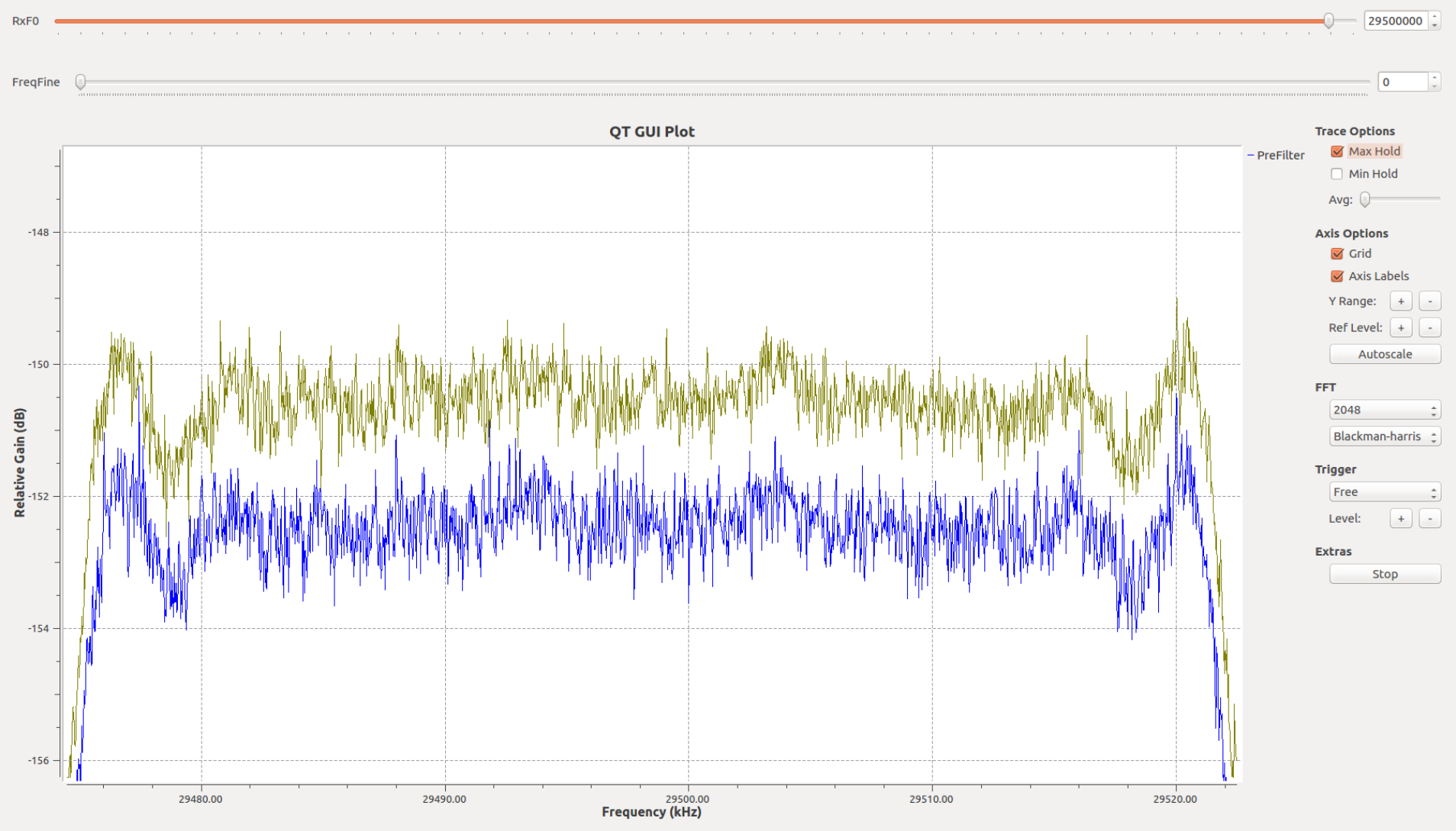
  - Radio: Open HPSDR / TAPR Hermes + Alex (filters & relays).
  - Noise Source: 5 dB Excess Noise Ratio (ENR).



Broadband
Calibrated
Noise Source



Hermes + Alex NF Measurement (gnuradio)



- 5 dB ENR Source ‘ON’ vs. Source ‘OFF’
- 48 Ks/s, 2048 FFT, bin size = 23.4 Hz.
- Exponential Averaging, tau ~ few seconds
- 2 dB / major division

- 3rd Demo – AM BCB Receiver.
  - Homodyne downconversion (to near zero)
    - The IQ Local Oscillator and mixer are in the radio hardware
  - Weaver method to select one sideband.
  - Filtering and AGC.
  - Output to a specific soundcard.

References

- https://wiki.gnuradio.org/index.php/Main_Page - Gnuradio Wiki
- https://openhpsdr.org/ - Open HPSDR website.
- https://redpitaya.com/ - Red Pitaya website.
  - Red Pitaya needs Pavel Demin's Red Pitaya OpenHPSDR firmware
  - plus N5EG's gr-hpsdr to talk to gnuradio (below).
- https://www.tapr.org/~n5eg – Echo sounding experiments (using gnuradio), links, other presentations.
- https://github.com/Tom-McDermott - GNU Radio drivers for OpenHPSDR, other source code.
- https://w7fu.com/ - John Petrich's "Ham-Friendly DSP" site.
- https://tapr.org/digital-communications-conference-dcc/ - ARRL / TAPR Digital Communications Conference (DCC). All things SDR and more.

Appendix

Gnuradio Throttle

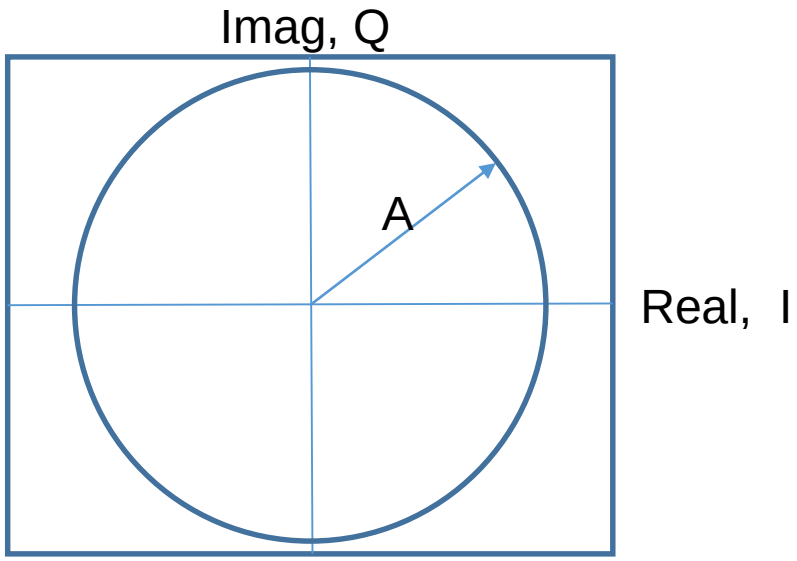
- A ‘throttle’ is needed *if* there is no source of physical timing.
  - A radio, or a soundcard (output or input) are physical sources of timing.
  - Simulated sources, files, all output samples as fast as the CPU can generate them.
    - This starves the computer, prevents servicing the GUI.
  - The throttle prevents the CPU from working at 100%.
  - GRC will warn you if it thinks you need one.
  - It can be in any data path. If needed, use only one throttle.
  - Windows audio sink can be a little goofy. Sometimes needs a throttle.

Complex vs. Real

- A complex signal contains both an In-Phase (I) signal and a Quadrature-Phase (Q) signal.
  - Real = one floating point number
  - Complex = a pair of floating point numbers.
  - Gnuradio keeps the complex number parts together as a pair.
- A complex number describes a vector on the complex plane.
  - CCW rotation once/sec = positive 1 Hertz.
  - CW rotation once/sec = negative 1 Hertz.
- A real number has no rotation – the vector exists only on the real axis.
  - Cannot differentiate negative frequency from positive frequency.
  - Therefore: it’s both frequencies at the same time.

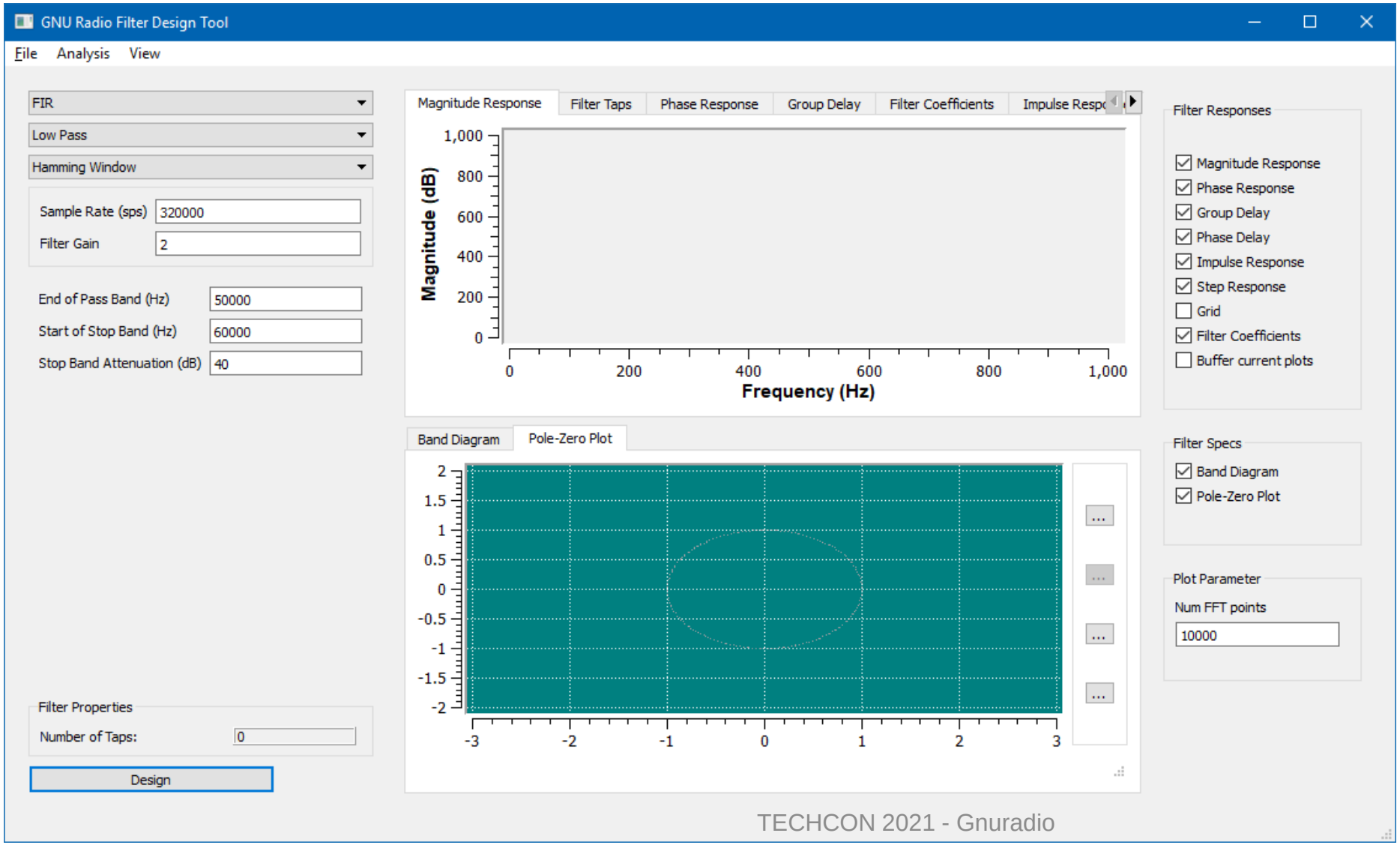
Complex notation

- Single frequency = A { cos (ωt) + i sin (ωt) } = A $e^{i\omega t}$
- Cos = I i Sin = Q A = Amplitude of the signal
- ω = frequency * 2π



Filter Designer

Tools → Filter Design



- Use to create custom filters.
- Creates ‘filter taps’ file that can be read by filter block.
- Standard filter blocks don’t need any of this.

Sample Rate & Decimation

- You must keep track of the sample rate throughout your flowgraph. Otherwise bad things happen.
  - Decimation & interpolation change the sample rate.
- Flowgraph creates a variable: samp_rate
  - You can use that name, or change it. You can make additional variables (for any purpose).
- Blocks that depend on the sample rate normally should use a variable.
  - Benefit: If you change the sample rate, then all blocks using that variable change along with it. Otherwise you have to hunt down and change all the rate-dependent blocks.

Decimation

- Normally the radio produces samples at too fast a rate – overwhelms CPU capability.
- Once you have isolated the frequencies of interest, frequency shift to zero Hertz, then decimate.
- Decimation reduces the sample rate by a factor of N.
- Nyquist criteria: You must low-pass-filter to ±Fs/2N or less before decimating.
  - Blocks downstream of the decimation thus process much fewer samples.
- Match sample rate between devices.
  - Example: Radio (perhaps 192 ksps) to Soundcard (perhaps 48 ksps).
    - LPF to less than 24 kHz (passband + roll off < 24k) Nyquist.
    - Then decimate by 192 / 48 (i.e. decimate by 4).