Get it on GitHub

Checkout the documentation

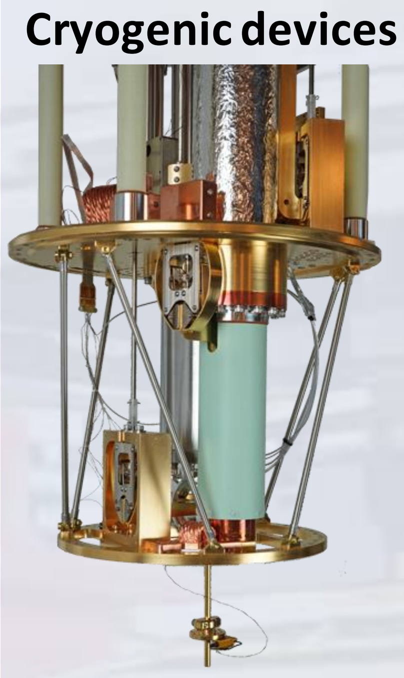
Jet Propulsion Laboratory
California Institute of Technology

A Flexible Radio-frequency readout for multiplexed Submillimeter-Wave Detectors

Lorenzo Minutolo (JPL/Caltech) Roger O'Brient (JPL), Bryan Steinbach (Caltech), Albert Wandui (Caltech), Katie Hughes (UChicago)

General scheme

Cryogenic devices

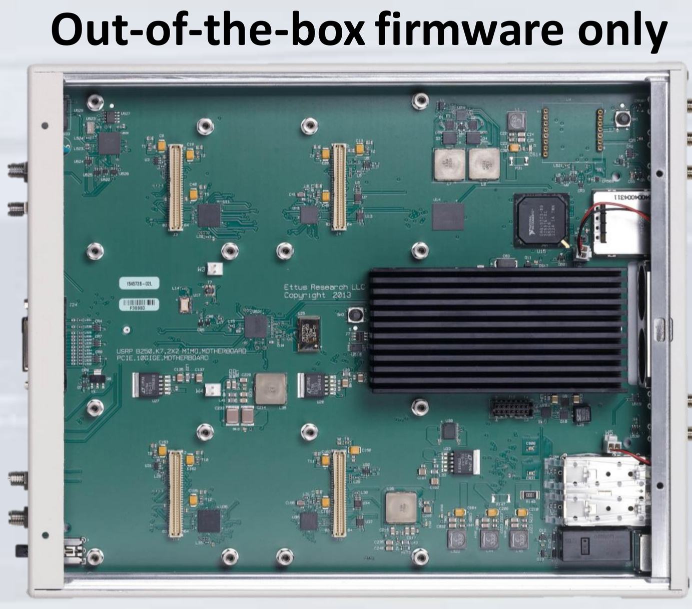


2x 160MHz BW TX/RX lines

Tunable between DC – 6 GHz

Integrated TX and RX amplifiers

Out-of-the-box firmware only



USRP SDR (here x300): Software defined radio: handles the baseband to device frequency shift, signal amplification and DAC/ADC.

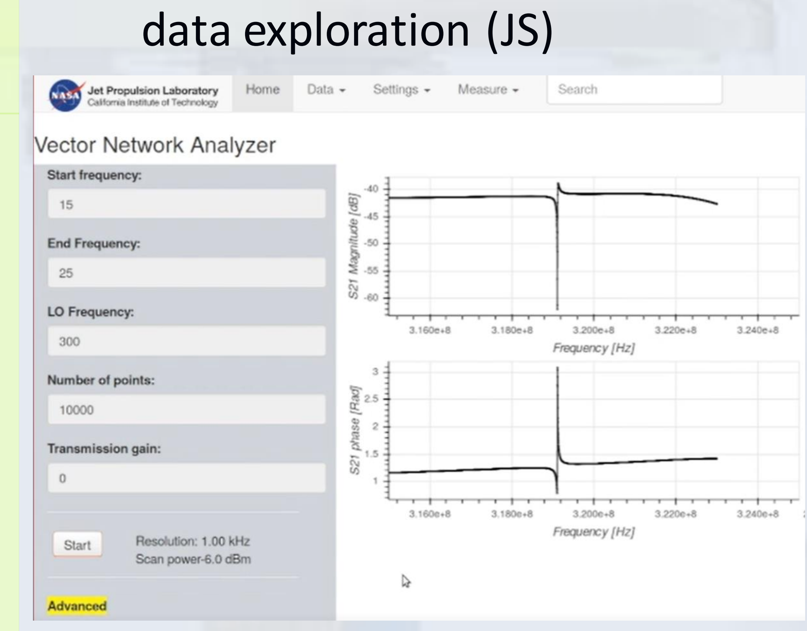
2x 10GBe links
or 1x 40GBe
or 1x USB3

GPU server: scale with your application: from a gaming computer to a multi-GPU server

PCIe x16/x8

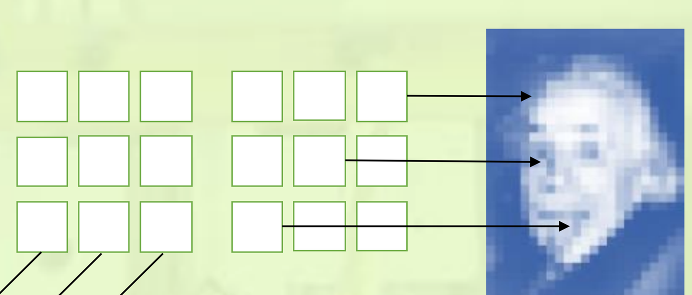
The link between server and client can be any connection supporting TCP protocol including loopback

Python API: Used to give commands to the server, live plotting, data analysis, storage (HDF5), publication level plotting, interactive data exploration (JS)

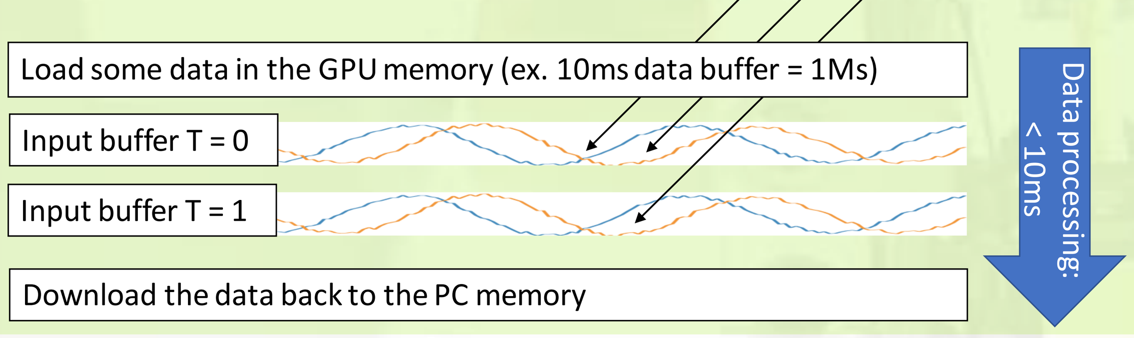


Sample of the web GUI

- The GPU coordinates $\propto 10^3$ "Slow" processors or **CUDA cores** on a single task
- The CUDA cores only operate on the GPU on-board memory. Memory transfer from the computer RAM to the GPU ram is required



GPUs have been developed to work with images: nowadays they work extremely fast on images: 2D grid of triplets



The same computing power can be used to **process samples**. The data processing in our software is faster than the data acquisition making the system **real-time**.

Nvidia GPU: Uses CUDA to do computationally demanding calculations. Instead of maintaining/developing a firmware, a GPU kernel can be programmed in CUDA (C)

Features

Stock features:

- **VNA scan:** measure, save and plot S21 functions with arbitrary resolution, IF bandwidth and **frequency sweep direction**
- **Multitone noise acquisition:** Generate and demodulate a com of tones. Tested up to 50k tones @ 2ksps each with -80dBc/Hz @ 1kHz
- **Signal generation:** generate baseband signals in real time, user defined signals or upmix external source.
- **Spectrum analyzer:** measure, saves and plot the entire spectrum at user defined intervals

Scalability:

- **Upgrading the GPU:** By replacing the GPUs the available computational power and memory available for calculations increase. The same code will handle for example more tones without any software modification.
- **Stacking more USRPs**: more USRPs can be connected to the same server and tuned to acquire adjacent region of the frequency spectrum

Customization:

The system is specifically designed to run new, **application specific** code. There are two level of customization (and a tutorial on each one):

- **CUDA** (GPU server): Work directly with the baseband buffers, implement real-time DSP, additional filtering, real time fitting and slowsignal feedback.
- **Python** (API client): implement new triggers, DSP on pre-processed signal (ex. Common noise subtraction), combine different DSP with different excitation signals.

Expected cost of the system

Readout server(excluding GPU): (200 MHZ – 400 MHz) 1k$ - 2.5k$

GPU unit: >0.5k$

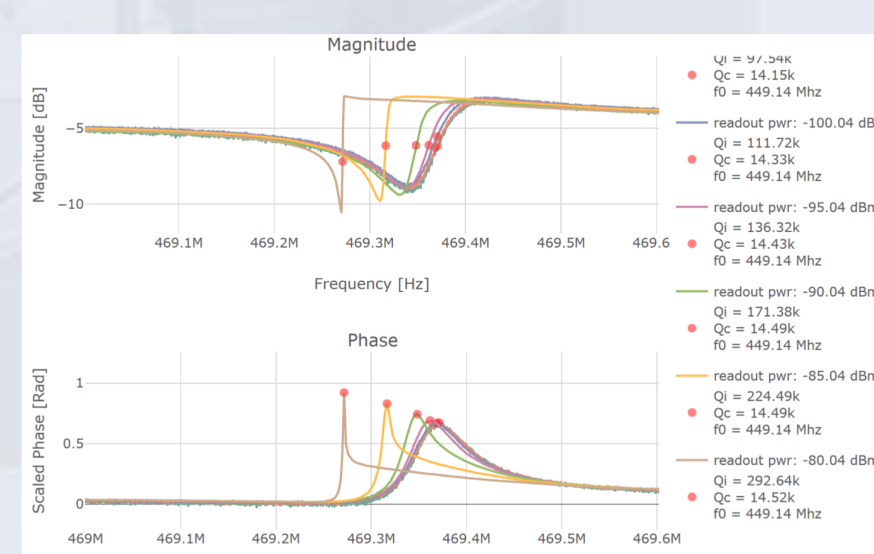
SDR: (full optional) 20k (standard) 10k (reduced BW) 5k

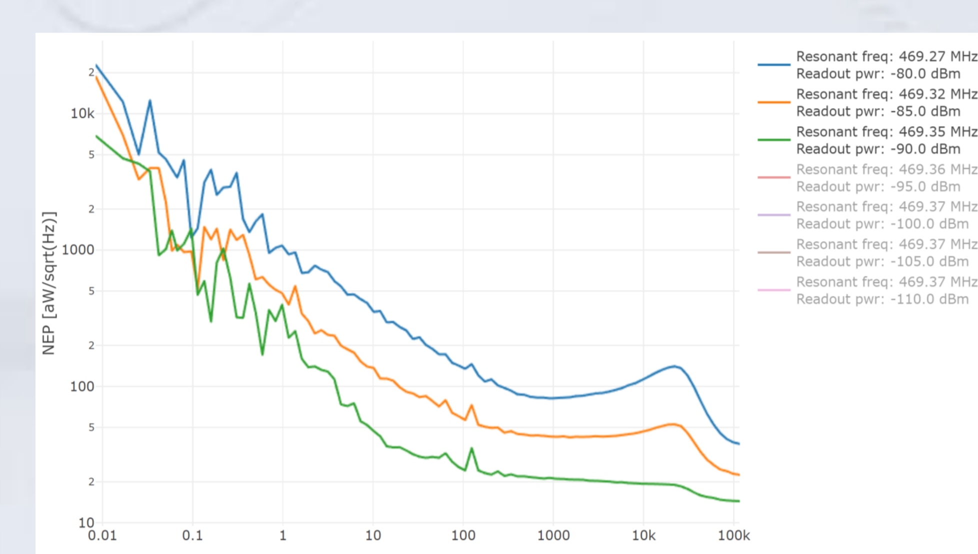
SOFTWARE: free, demonstrated and all opensource!

TOTAL (base config – high end config) 6.5k$ – 24k$

Applications

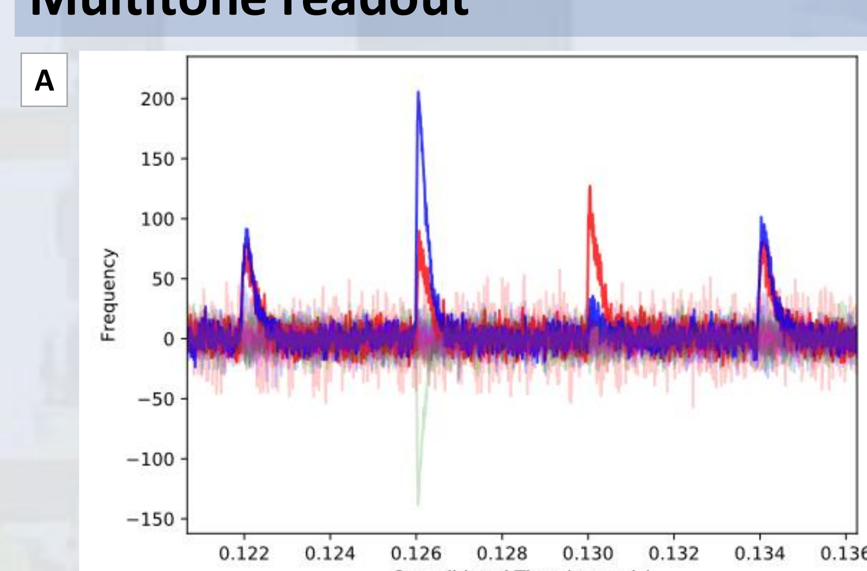
Devices characterization



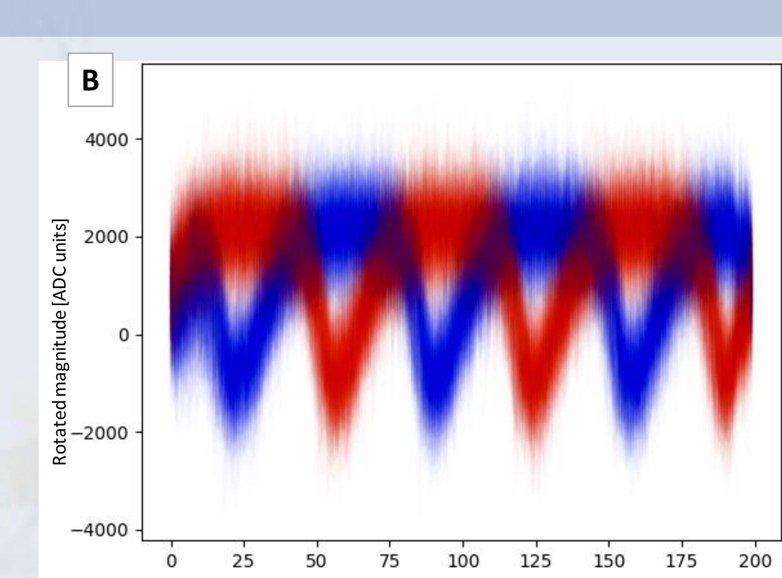


Multiple VNA scan taken variating the readout power. After each VNA the resonator is fitted and the best readout point (red dots) is determined. A noise acquisition targeting that point is saved to disk and analyzed. Using this technique we determined the best noise performance of **TKIDs** (**see Albert Wandui talk**). The measurement process (including on-board gain variation and fitting) is automated for maximum reproducibility.

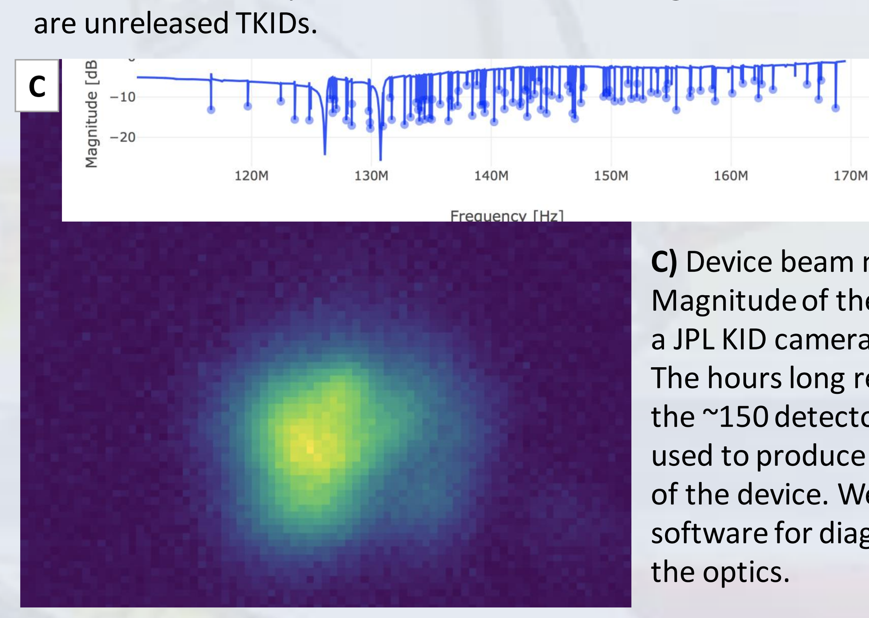
Multitone readout



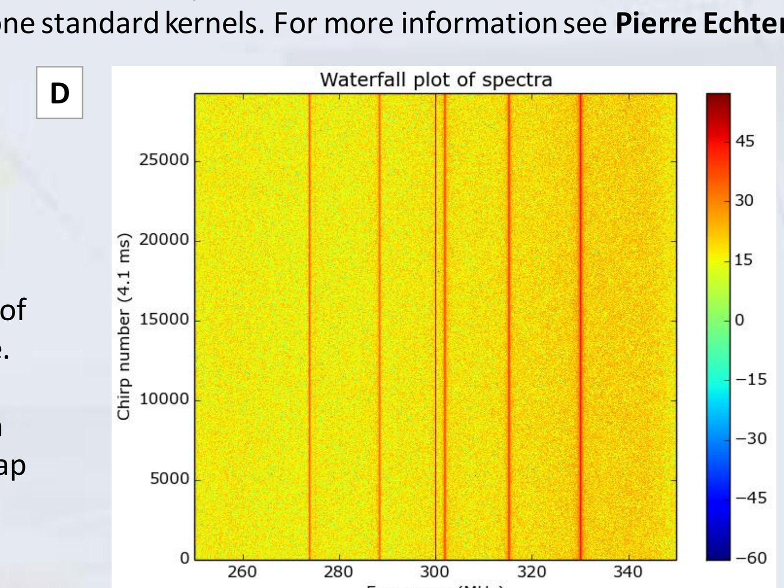
A) Triggered multi-channels acquisition: 14 channels acquired at 200ksps. The threshold trigger only saves the data of excitation events. The trigger is coded in the Python API and operates in real-time on already decimated data. The investigated devices are unreleased TKIDs.



B) Eye diagram of a single **QCD** (Quantum capacitance detector) trace. When the QCD absorb a photon the resonance oscillation due to the bias ramp changes phase. The ability to distinguish between the two phases has been implemented in the Python API while the GPU server has been used with the multitone standard kernels. For more information see **Pierre Echternach** talk.



C) Device beam mapping: Magnitude of the S21 scan of a JPL KID camera prototype. The hours long readout of the ~150 detector has been used to produce a beam map of the device. We used this software for diagnostic of the optics.



D) Fast chirp readout: a 4.1 ms long chirp signal swipe the frequency region the resonators are in storing energy in each of them. The subsequent energy release is captured and the process is repeated. The plot does not show the chirp signal but only the resonators energy release.