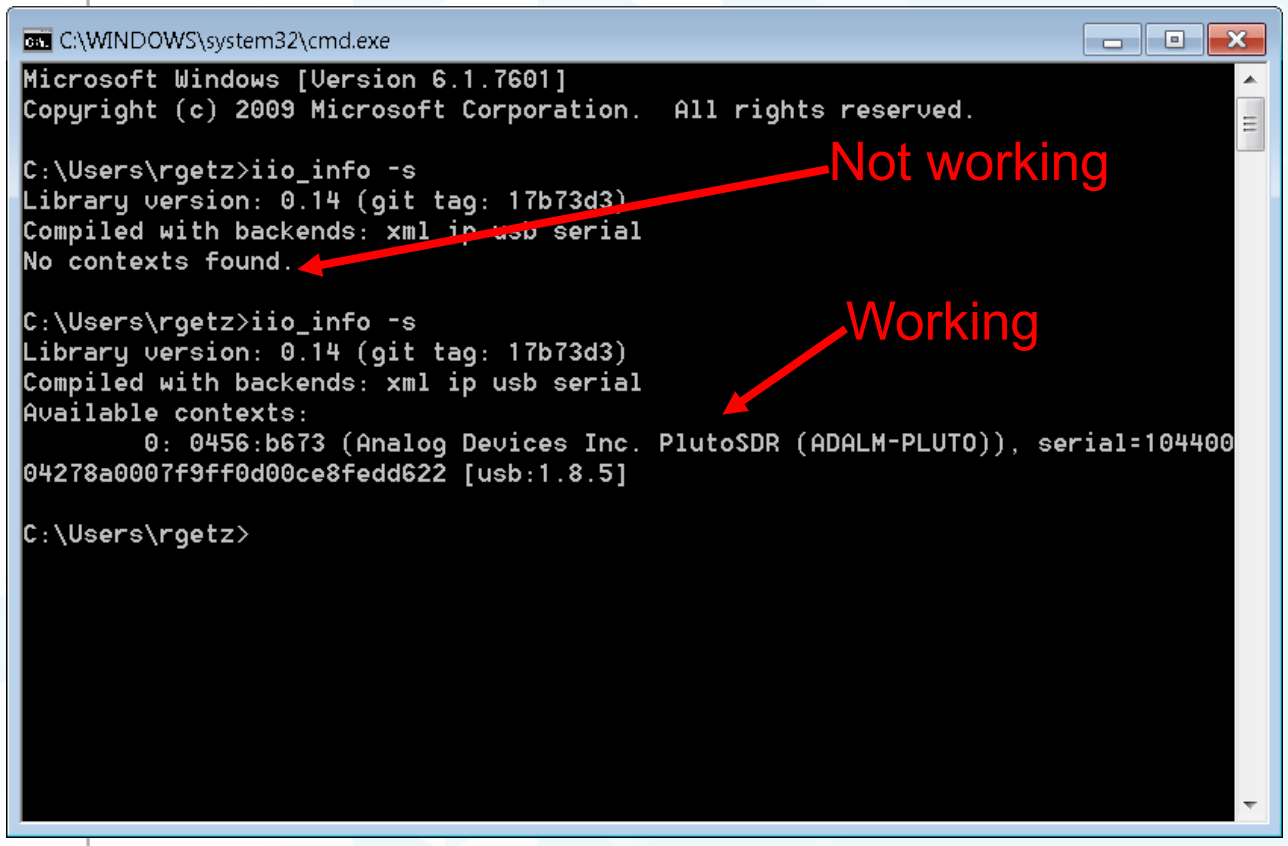
Checking out your hardware/software

Try this out now! Help us help you.

- From console/DOS window type:
  - "**iio_info -s<ret>**"
  - <ret> indicates return key
  - iio_info (underscore), not iio-info (dash)
  - Should find the PlutoSDR
    - If this works – great
    - If this doesn't work – please put up your hand, and someone will come to help



CONNECT LOOPBACK CABLE

Get the labs (if you haven't already)

- Available at
  - https://github.com/sdrforengineers/LabGuides/tree/master/grcon2020

Hands-on Workshop: Introduction to ADALM-PLUTO and IIO

Robin Getz

Travis Collins, PhD

AHEAD OF WHAT'S POSSIBLE™

What is an SDR?

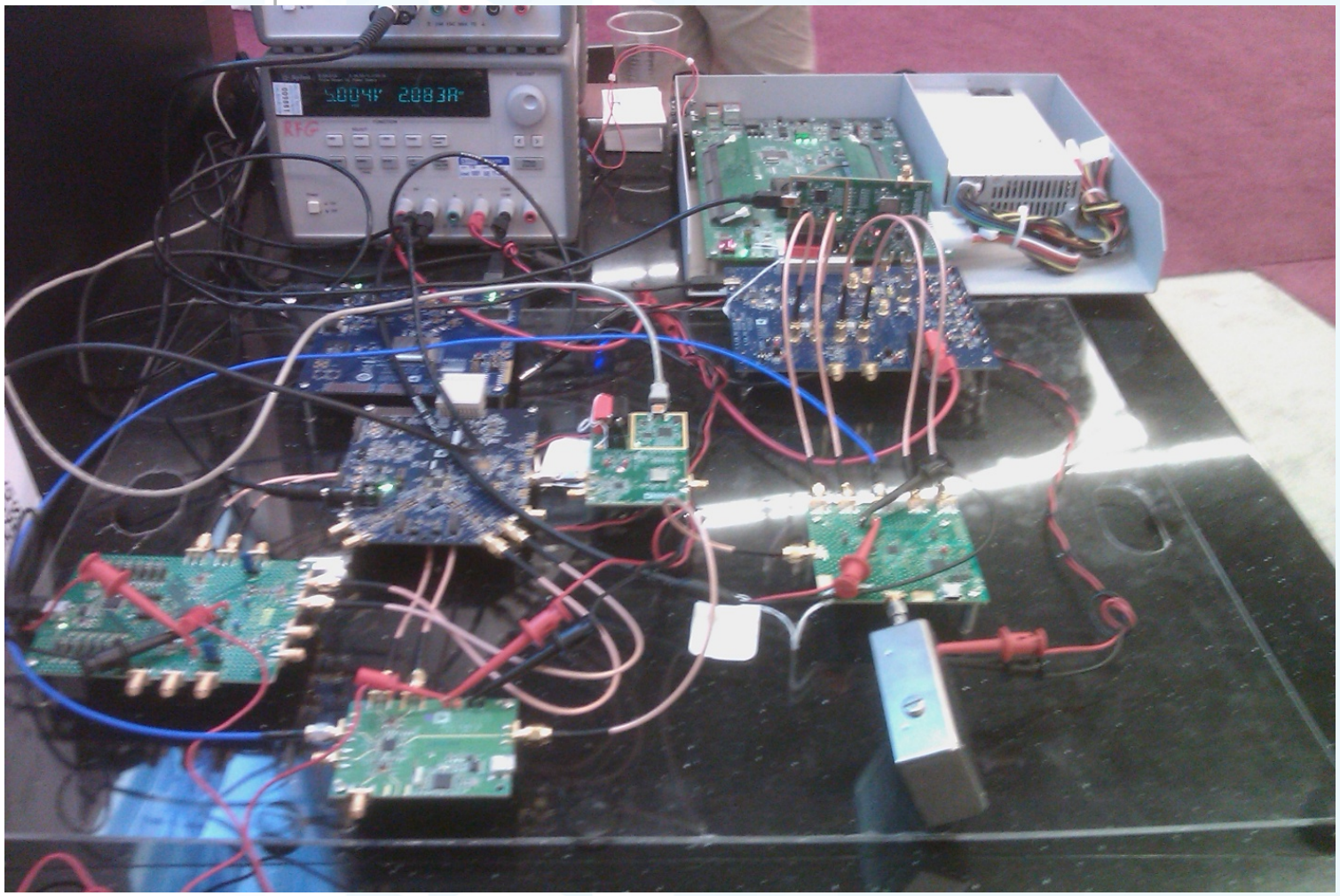
- ***Software-defined radio (SDR)*** is a radio communication system where components that have been traditionally implemented in hardware (e.g. mixers, filters, amplifiers, modulators/demodulators, detectors, etc.) are instead implemented by means of software on a personal computer or embedded system.
- While the concept of SDR is not new, the rapidly evolving capabilities of digital electronics render practical many processes which were once only theoretically possible.

- `https://en.wikipedia.org/wiki/Software-defined_radio`

Traditional RF Evaluation Platforms

(Antenna to Bits and Back™, circa IMS 2010)

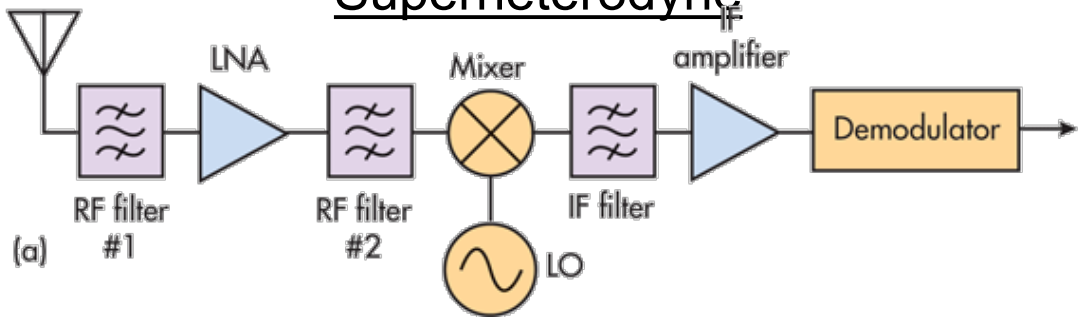
- Discrete single product evaluation boards,
  - connected with SMA cables
  - Power supplies
  - External oscillators
  - Tuning filters
  - No Antenna
- 6 power supplies
- 4 different USB applications

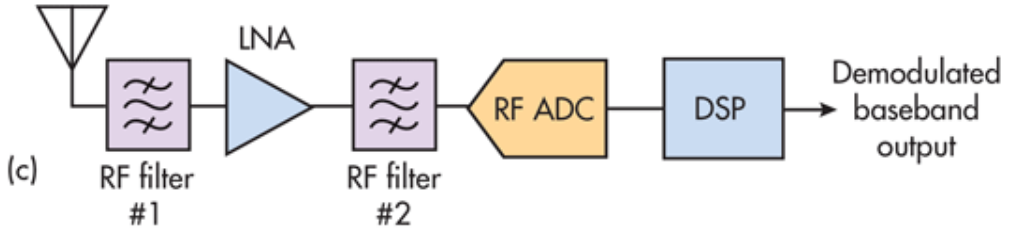


- Not easy to replicate, or use as part of a SDR prototyping solution
- Needed small form factor, open design

Basics: Radio Architectures

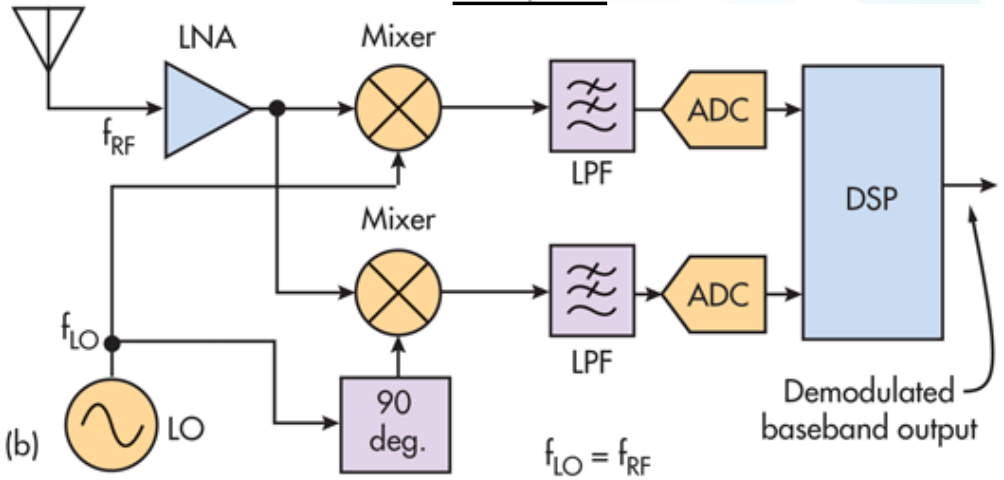
Superheterodyne





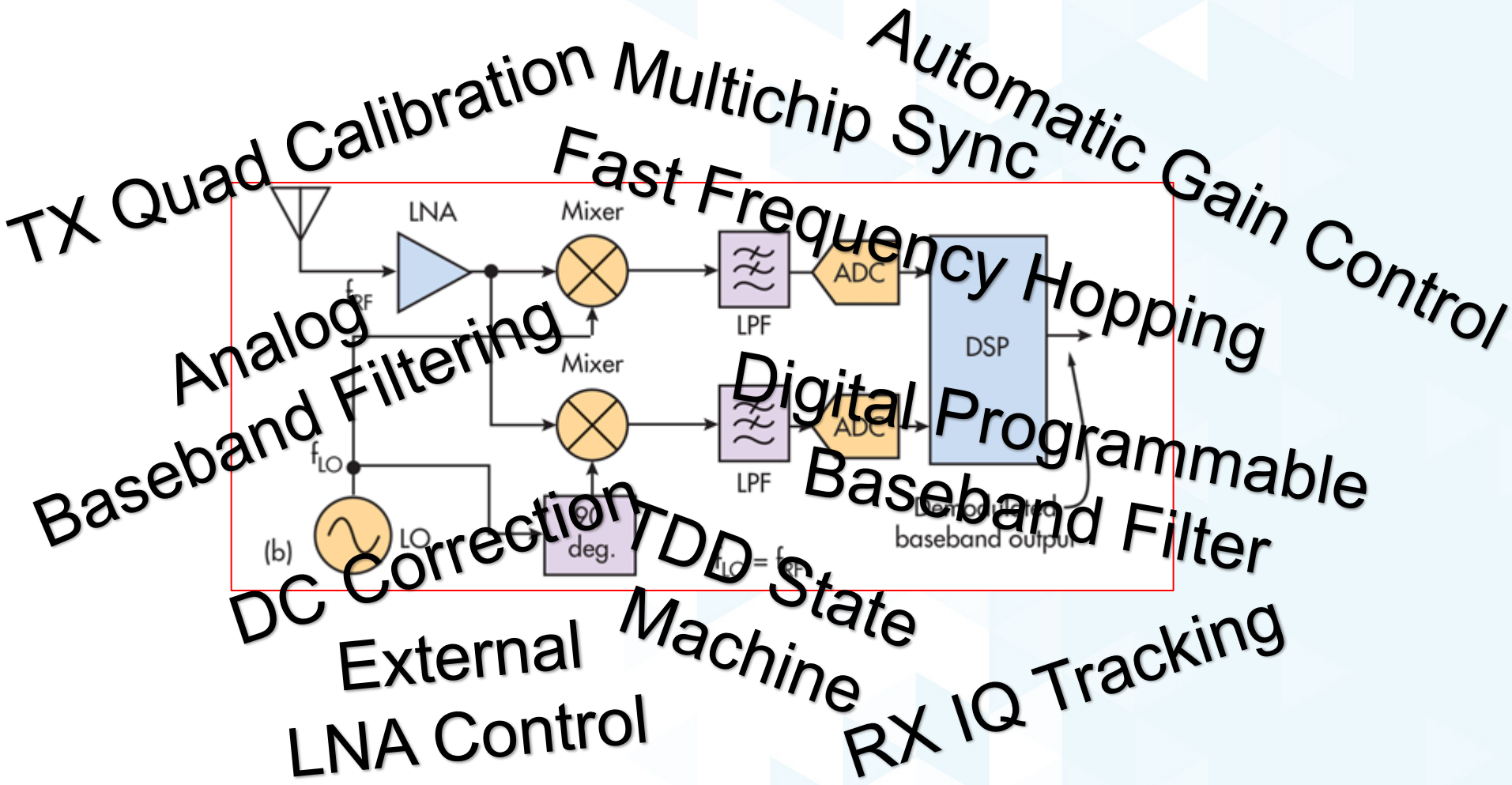
Zero IF

Direct RF

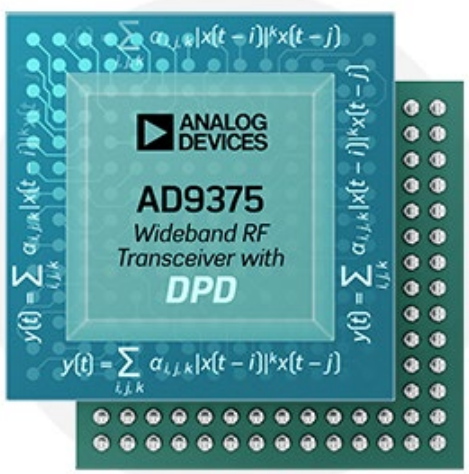


https://www.electronicdesign.com/adc/high-speed-rf-sampling-adc-boosts-bandwidth-dynamic-range
Lou Frenzel | Feb 22, 2017

Zero IF : Not just a mixer + ADC/DAC



Transceiver Family

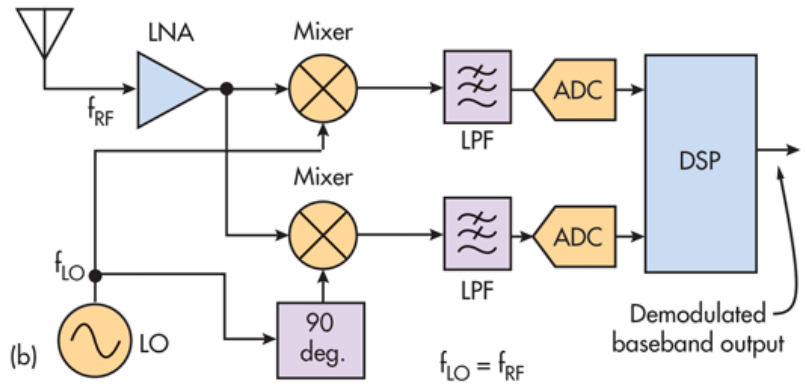


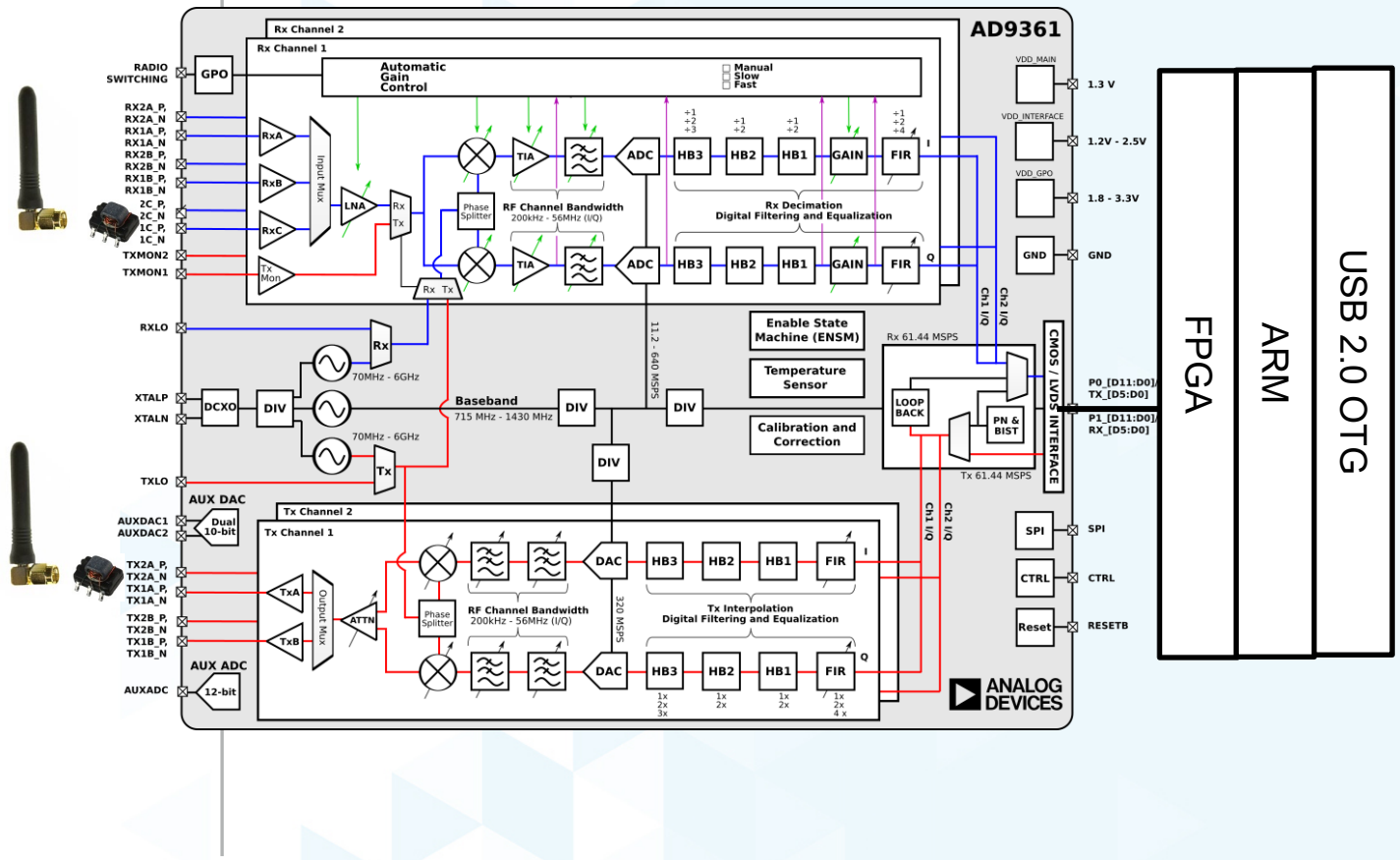
Transceiver Family

	Part #	Applications	Bandwidth	Functionality	RF Tuning Range	Rx Image Rejection*	Rx NF/IIP3**	Tx OIP3*	EVM	Package Size	Data Interface	DPD	Power Consumption
	**AD9363**	3G/4G Femtocell, UAV, Wireless Surveillance	200 kHz – 20 MHz	2 Rx (or Orx), 2 Tx	325 MHz to 3.8 GHz	50dB	3dB/-14dBm	+19dBm	-34 dB	10 × 10 mm 144-CSP_BGA 0.8 mm pitch	CMOS/LVDS	N/A	1.3 W
	**AD9364**	3G/4G Picocell, SDR	200 kHz – 56 MHz	1 Rx (or Orx), 1 Tx	70 MHz to 6 GHz	50dB	3dB/-14dBm	+19dBm	-40 dB	10 × 10 mm 144-CSP_BGA 0.8 mm pitch	CMOS/LVDS	N/A	0.8 W
	**AD9361**	3G/4G Picocell, SDR, Pt-Pt, Satcom, IoT Aggregator	200 kHz – 56 MHz	2 Rx (or Orx), 2 Tx	70 MHz to 6 GHz	50B	3dB/-14dBm	+19dBm	-40 dB	10 × 10 mm 144-CSP_BGA 0.8 mm pitch	CMOS/LVDS	N/A	1.3 W
	**AD9371**	3G/4G Macro BTS, Massive MIMO, SDR	100MHz Rx, 250MHz Tx/Orx	2Tx, 2Rx Orx & SnRx	300 MHz to 6GHz	75dB	13.5dB/+22dBm	+27dBm	-40 dB	12 × 12 mm 196-CSP_BGA 0.8 mm pitch	6.144 Gbps JESD204B	N/A	4.86 W
	**AD9375**	3G/4G Small Cell, 3G/4G Massive MIMO	100MHz Rx, 250MHz Tx/Orx	2Tx, 2Rx Orx & SnRx	300 MHz to 6GHz	75dB	13.5dB/+22dBm	+27dBm	-40 dB	12 × 12 mm 196-CSP_BGA 0.8 mm pitch	6.144 Gbps JESD204B	Linearization BW up to 40MHz	4.86 W
	**ADRV9009**	Macro BTS, Massive MIMO, Active Antenna, Phased Array Radar, Portable Test Equipment	200MHz Rx, 450MHz Tx/Orx	2 Rx (or 2 Orx), 2 Tx	75MHz to 6GHz	75dB	12dB/+15dBm	+27dBm	-43 dB	12 x 12 mm 196-CSP_BGA 0.8 mm pitch	12 Gbps JESD204B	N/A	5.66 W

* typical performance @ 2.6GHz
** typical performance @ 2.6GHz, AD9361 assumes internal LNA; AD937x and ADRV9009 no internal LNA.

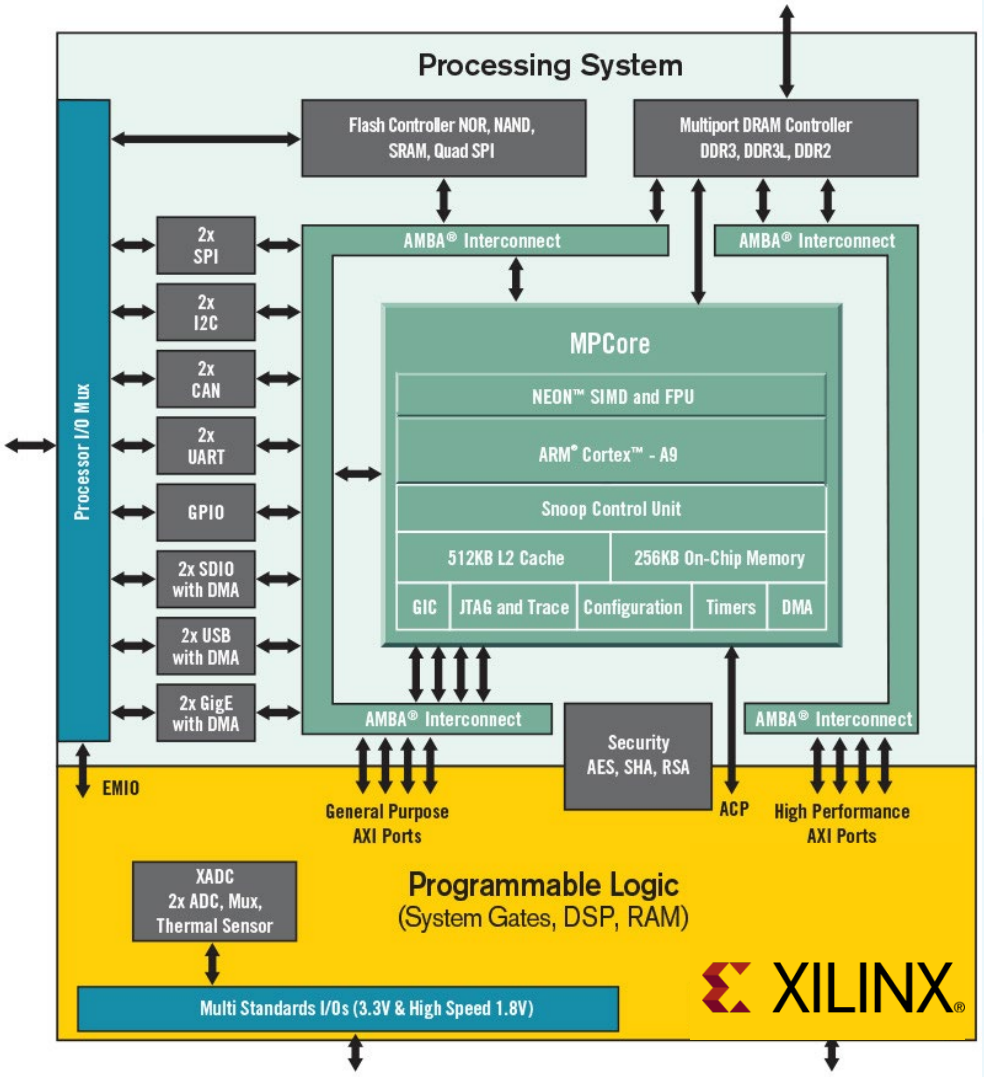
Zero IF == ADALM-PLUTO SDR





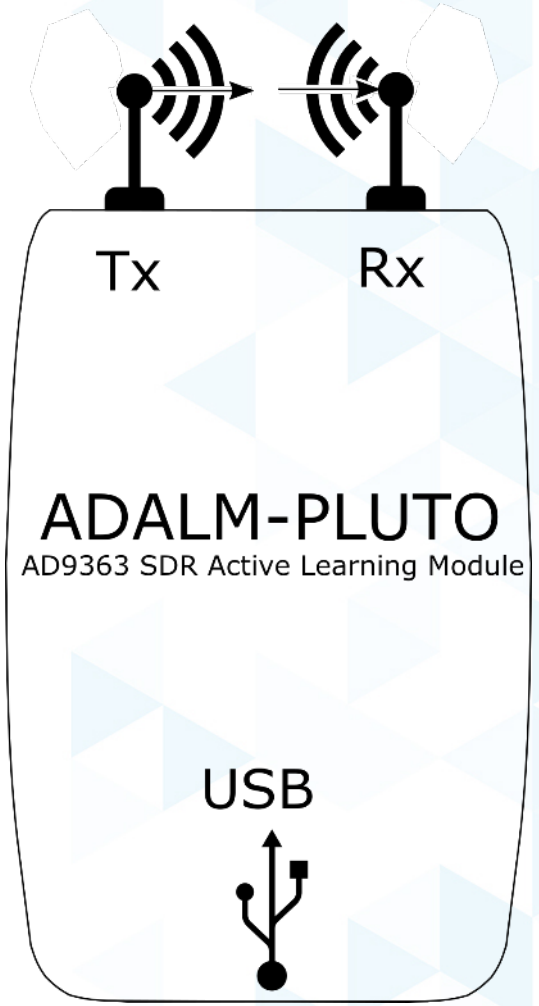
Xilinx Zynq

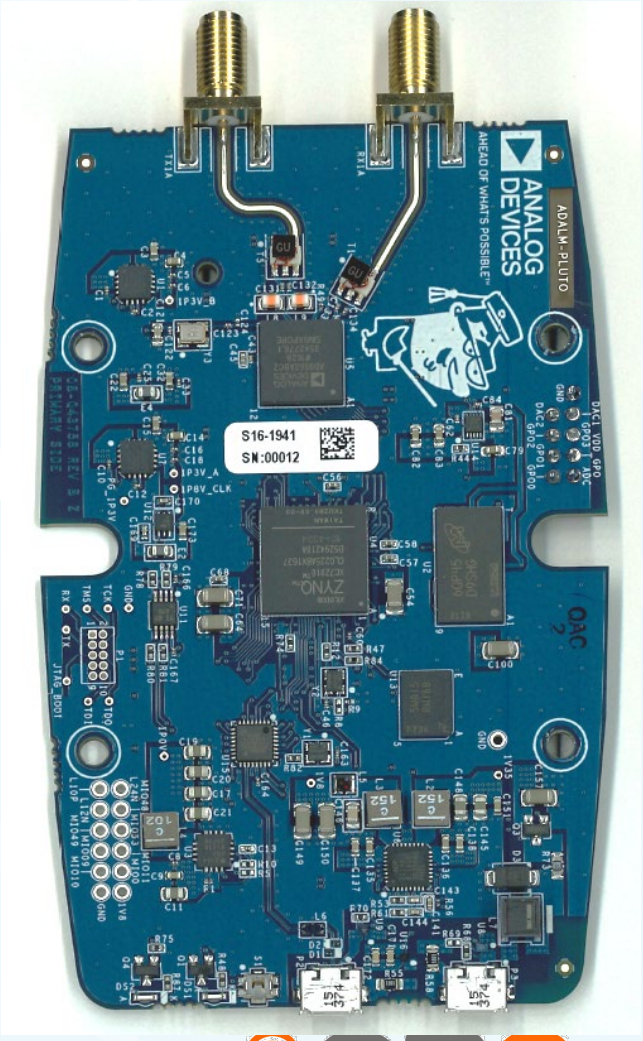
- Single Core Zynq 7010-4334
  - ARM Cortex-A9 MPCore (666.6 MHz)
  - NEON™ SIMD Engine
  - Single/Double Precision Floating Point Unit
  - USB 2.0 OTG
  - SPI
- FPGA
  - 28K Logic Cells
  - 17,600 LUTs
  - 35,200 Flip Flops
  - 2.1Mb
  - 80 DSP Slices
- Single CLG225 (13x13) 0.8 mm pitch
  - 32 MIO pins
  - 16 DDR data
  - 4 pairs of XADC signals
  - 54 FPGA High Range Pins (1.2V to 3.3V)



https://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html

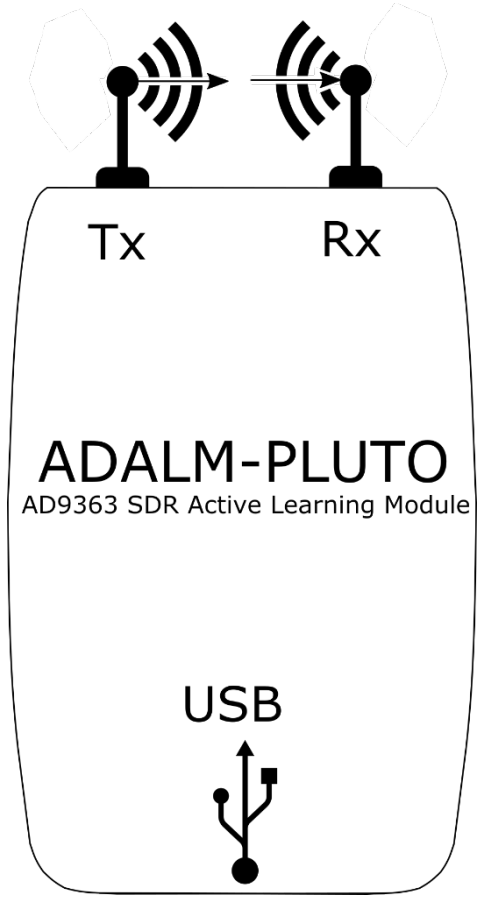
ADALM-PLUTO (PlutoSDR)





Newest Kit for students: ADALM-PLUTO

AD9363 Software Defined Radio Active Learning Module

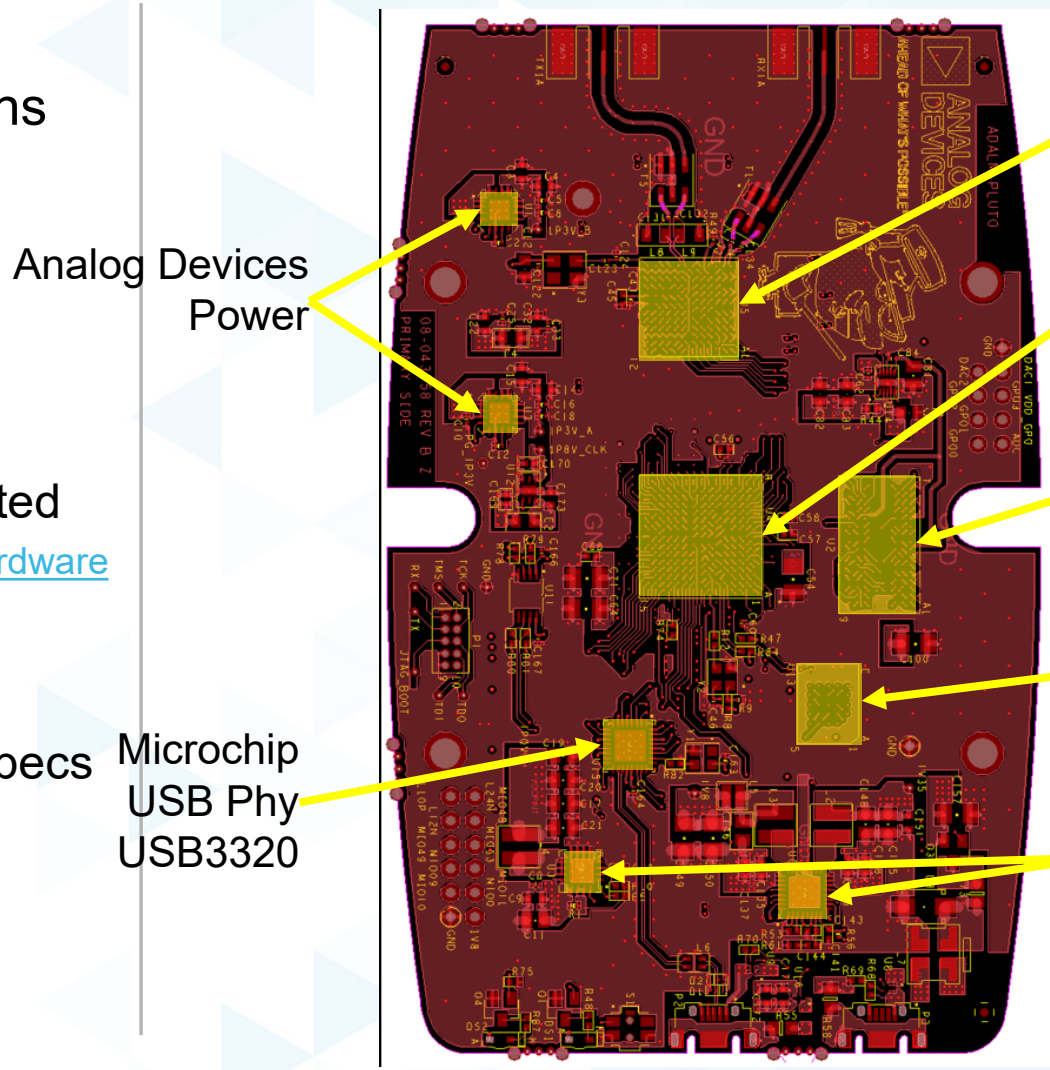


- Captures I/Q Samples
  - 12-bits
  - 65.1 kSPS to 61.44 MSPS
  - 200kHz to 20 MHz signal bandwidth
- Sends them to PC for processing over USB2

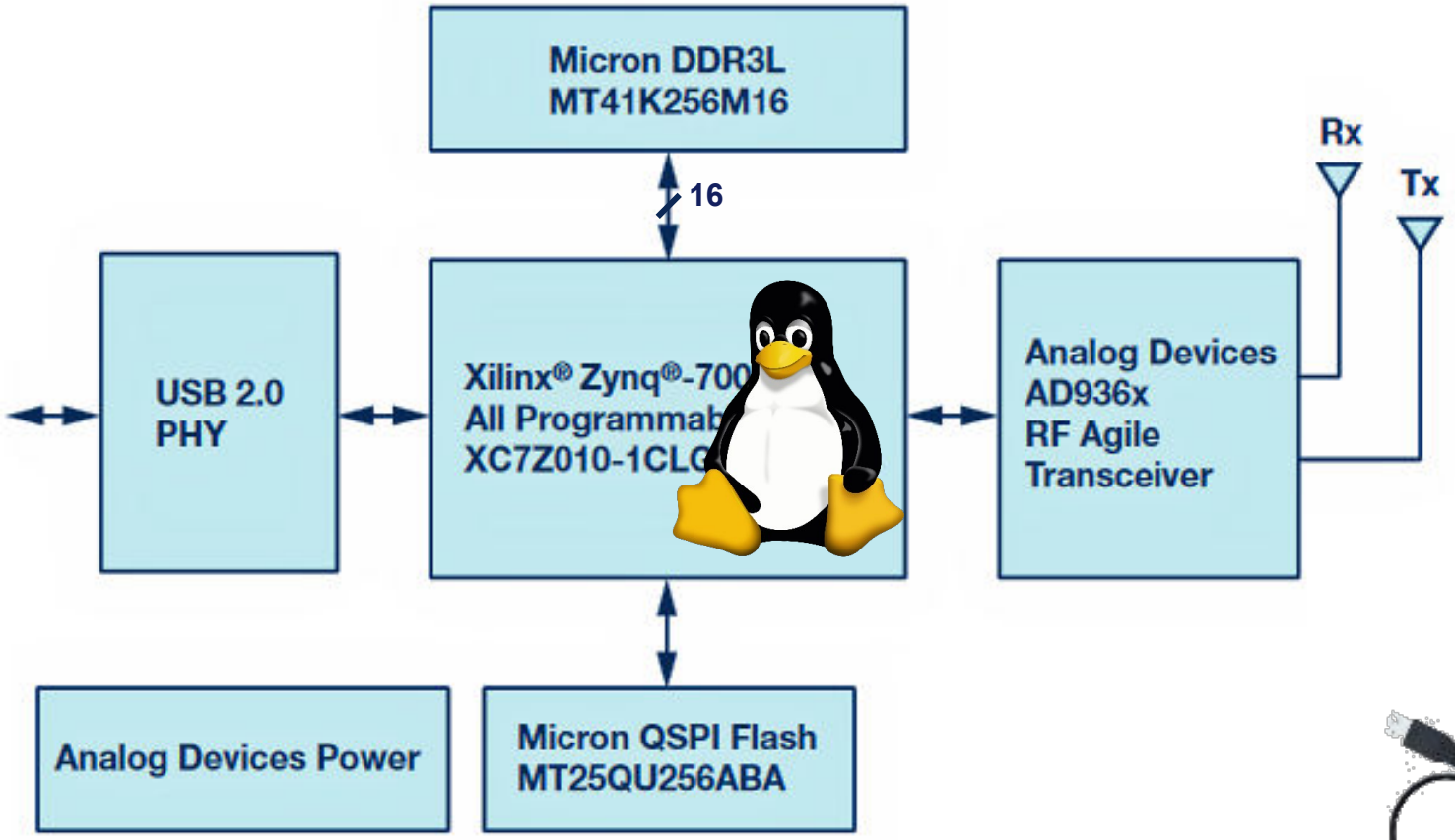
- $149
- Tuning range;
  - 325 MHz to 3.8 GHz
    - Guaranteed performance
  - 70 MHz to 6.0 GHz
    - Unknown specs

ADALM-PLUTO Design

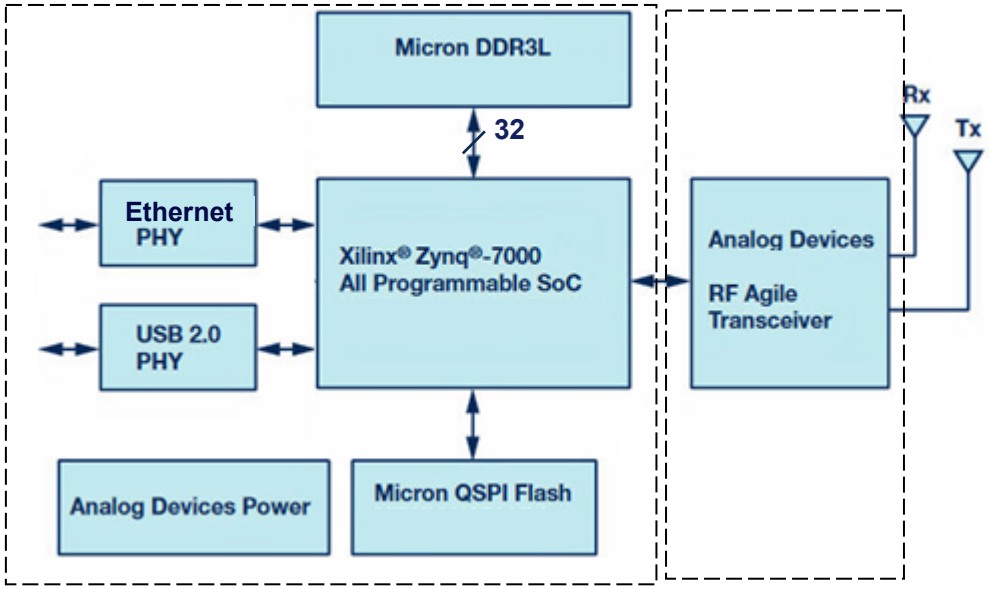
- Design is open, just like all other ADI designs
  - Shows a minimal full system design
    - From antenna to USB
    - RF to bits
  - Only 72 parts on the BOM
    - All IC, R, C, L, connectors, etc
  - Schematics, Gerbers, BOM, Allegro Files posted
    - https://wiki.analog.com/university/tools/pluto/hacking/hardware
  - Passes FCC and CE tests
  - Achieves better RF than AD9363 datasheet specs

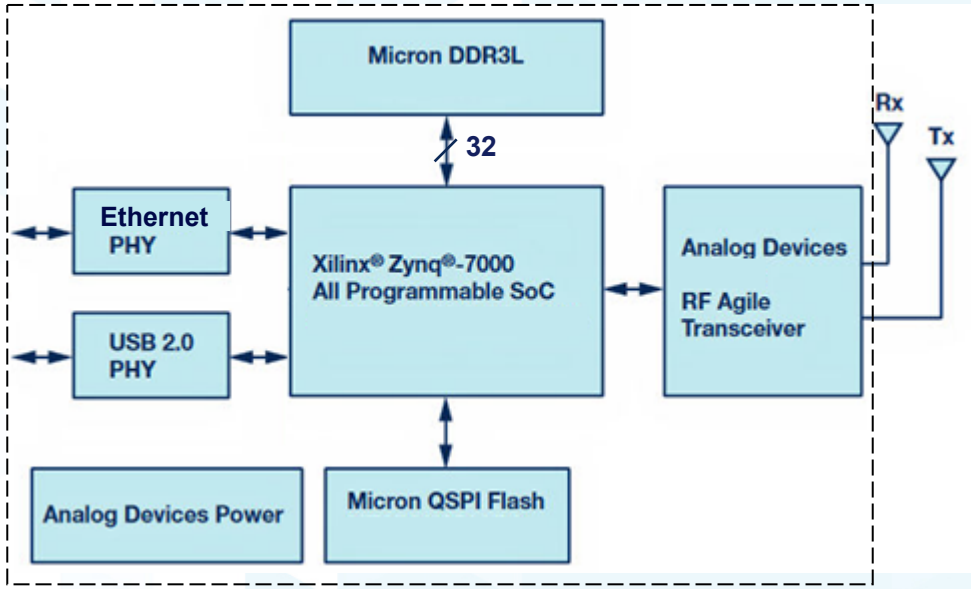


PlutoSDR: Block Diagram



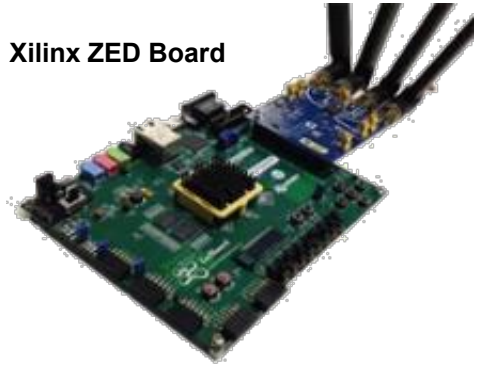
SDR Hardware Block Diagram





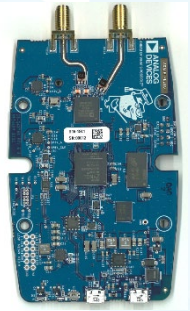
ADI ADFMCOMMS3-EBZ

Xilinx ZED Board



Epiq Solutions SideKiq Z2

ADALM-PLUTO

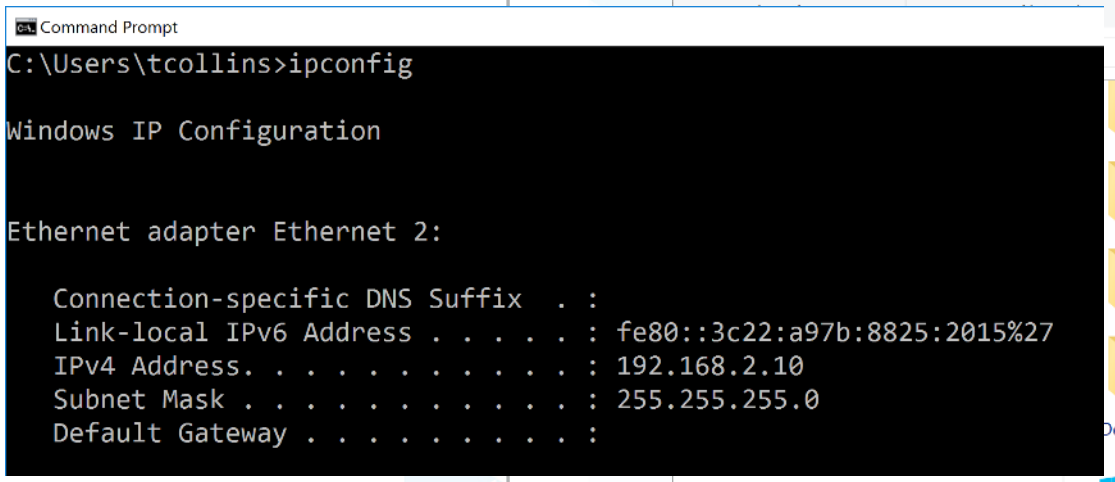


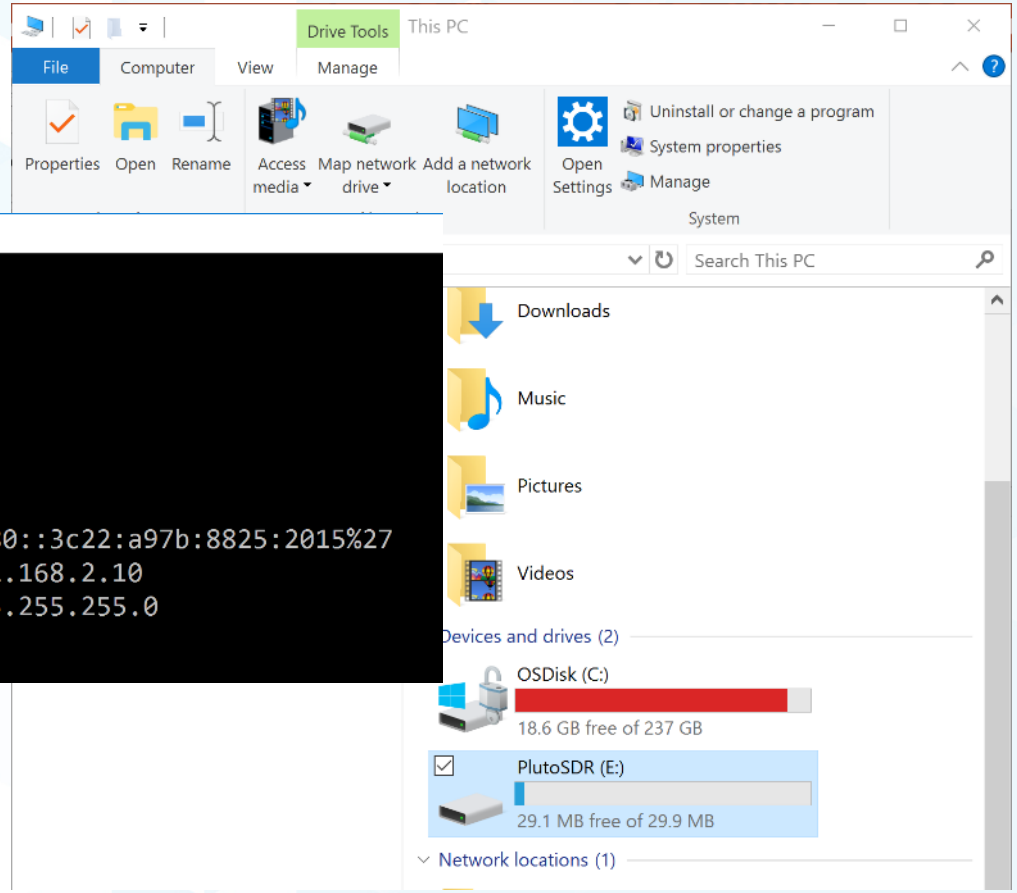
ADRV9361-Z7035
ADRV9364-Z7020

ADRV9009-ZU11EG

Connecting With PlutoSDR

- Pluto enumerates four devices
  - Mass storage
  - Ethernet (RNDIS)
  - Serial
  - IIO





Questions about Pluto SDR

- Pluto is an embedded Linux device
  - Uses Linux's USB Gadget
    - Mass storage
    - Serial
    - Ethernet (RNDIS)
    - IIO
  - Can be standalone
    - ~ same performance as Raspberry Pi 1
    - Single Core 666 MHz ARM A9
    - NEON + Floating Point
  - Supports USB Host
  - Open Source Firmware, Open Source HDL, Open Schematics
- For more information:
  - www.analog.com/plutosdr
  - wiki.analog.com/plutosdr
  - For further information, check out the text
    - Chapter 5: Understanding SDR Hardware

http://www.iconarchive.com/show/noto-emoji-objects-icons-by-google/62807-radio-icon.html
http://www.streamlineicons.com
http://pixelkit.com

USB OTG

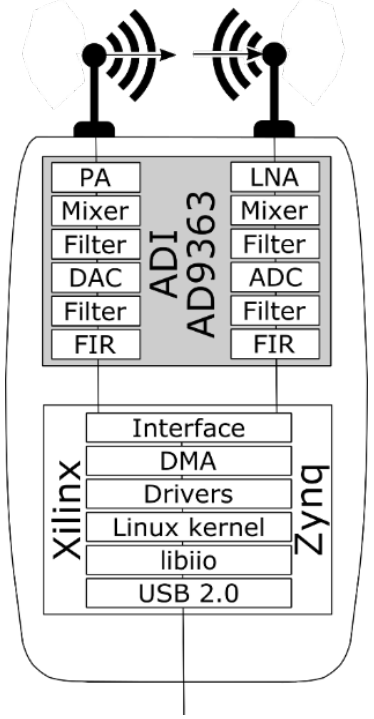
- The Pluto will automount any USB mass storage device such as thumb drive or Hard Drives
- The automounter will then look for some special file names:
  - **`runme[0-9].sh`** which it will run as a shell script
  - **`runme[0-9]`** which it will run as a binary file.
- For those interested, it will do that via the automounter script in **`/lib/mdev/automounter.sh`**

1. Provide power to right USB
2. Plug OTG+USB drive into left USB port
3. Left LED will become solid
4. Script runs

ADALM-PLUTO USB OTG Connectivity Options

ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

Connect to host



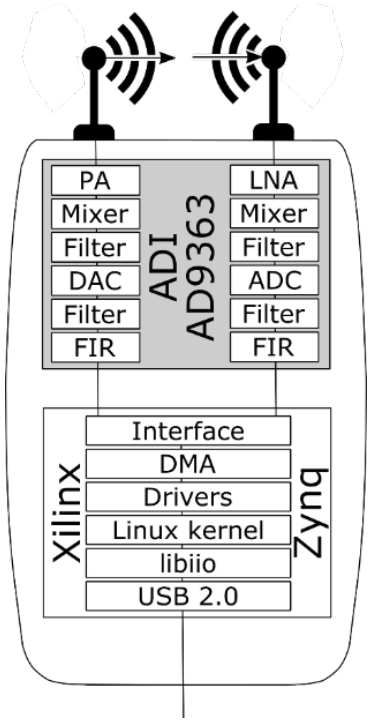
Linux

Mac

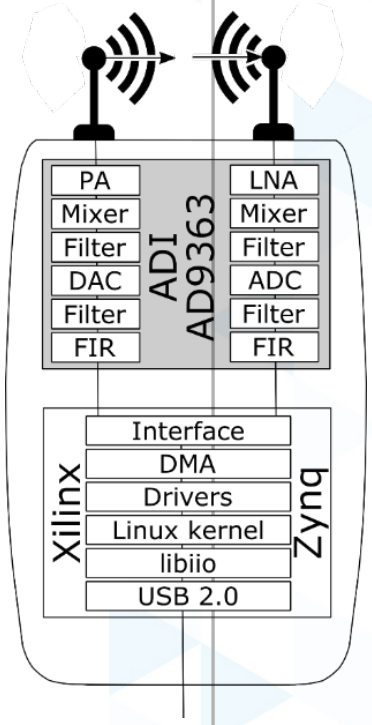
Windows

eLinux

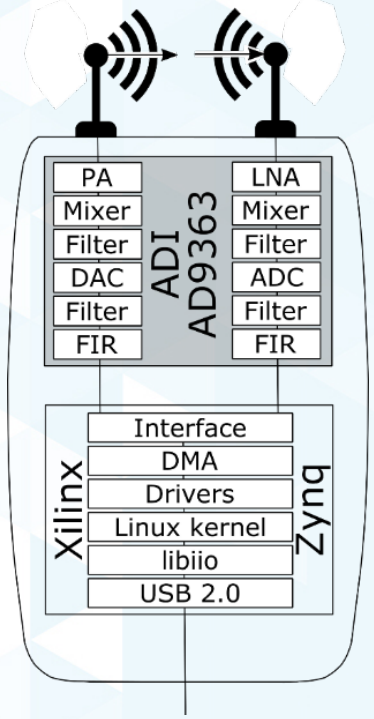
USB Thumb Drive



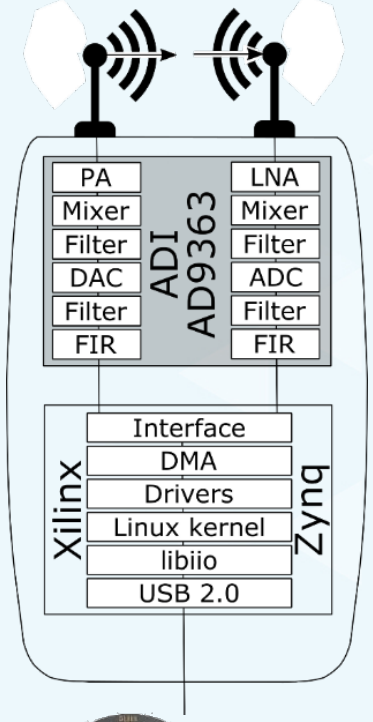
USB LAN



USB WiFi



USB Audio



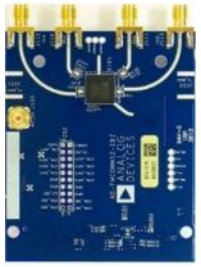
GRCon

Evaluation and Prototyping Hardware

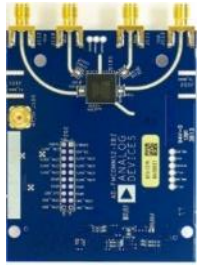
ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

ADALM-PLUTO
- AD9363
- 1 x Rx, 1 x Tx
- *325 MHz – 3.8GHz*
- 200kHz – 20 MHz channel bandwidth

AD-FMCOMMS2
AD-FMCOMMS3
- AD9361
- 2 x Rx, 2 x Tx
- *tuning range*
  - *2.2 GHz – 2.6GHz*
  - *70 MHz – 6GHz*
- 200kHz - 56 MHz channel bandwidth

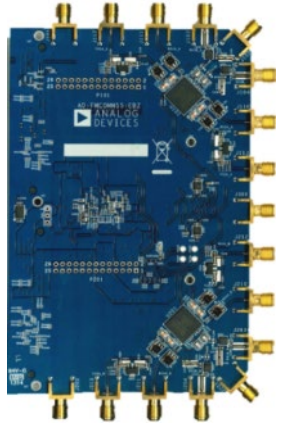


AD-FMCOMMS4
- *AD9364*
- *1 x Rx, 1 x Tx*
- 70 MHz – 6GHz tuning range
- 200kHz - 56 MHz channel bandwidth
- Shipping Now



ARRADIO
- AD9361
- HSMC, not FMC
- 2 x Rx, 2 x Tx
- *2.2 GHz – 2.6GHz tuning range*
- 200kHz - 56 MHz channel bandwidth
- Shipping Now!

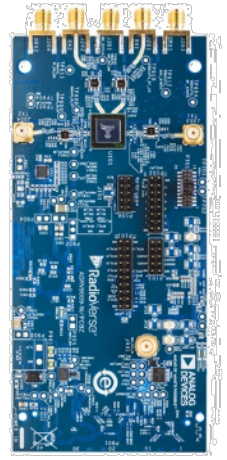
AD-FMCOMMS5
- *2 x AD9361*
- *4 x Rx, 4 x Tx*
- *Synchronized RF*
- 70 MHz – 6GHz tuning range
- 200kHz - 56 MHz channel bandwidth
- Shipping Now!



ADRV9371-N/PCBZ
ADRV9371-W/PCBZ
- *AD9371*
- *2 x Rx, 2 x Tx, 2 x Obs, 1x Sniffer*
- tuning range
  - 1.8GHz – 2.6GHz
  - 300MHz – 6GHz
- Tx synthesis bandwidth 250 MHz
- Rx BW: 8 MHz to 100 MHz

ADRV9375-N/PCBZ
ADRV9375-W/PCBZ
- *AD9375*
- *2 x Rx, 2 x Tx, 2 x Obs, 1x Sniffer*
- tuning range
  - 1.8GHz – 2.6GHz
  - 300MHz – 6GHz
- DPD actuator and adaptation engine for PA linearization

ADRV9008-1W/PCBZ (Rx)
ADRV9008-2W/PCBZ (Tx/Obs)
ADRV9009-W/PCBZ (TDD)
- *ADRV9008-1, ADRV9008-2, ADRV9009*
- *2 x Rx, 2 x Tx, 2 x Obs, 1x Sniffer*
- 75MHz - 6GHz tuning range
- Tx synthesis bandwidth 450 MHz
- Rx BW to 200 MHz



ADRV9364-Z7020
ADRV9361-Z7035
- *AD9364 + Zynq 7020*
- *AD9361 + Zynq 7035*
- 70 MHz – 6GHz tuning range
- 200kHz - 56 MHz channel bandwidth
- 1GB DDR + 32MB FLASH
- Ethernet + USB Phy

PACKRF
- *ADRV9361 reference design*
- Battery, PoE, Screen, Audio, GPS, IMU

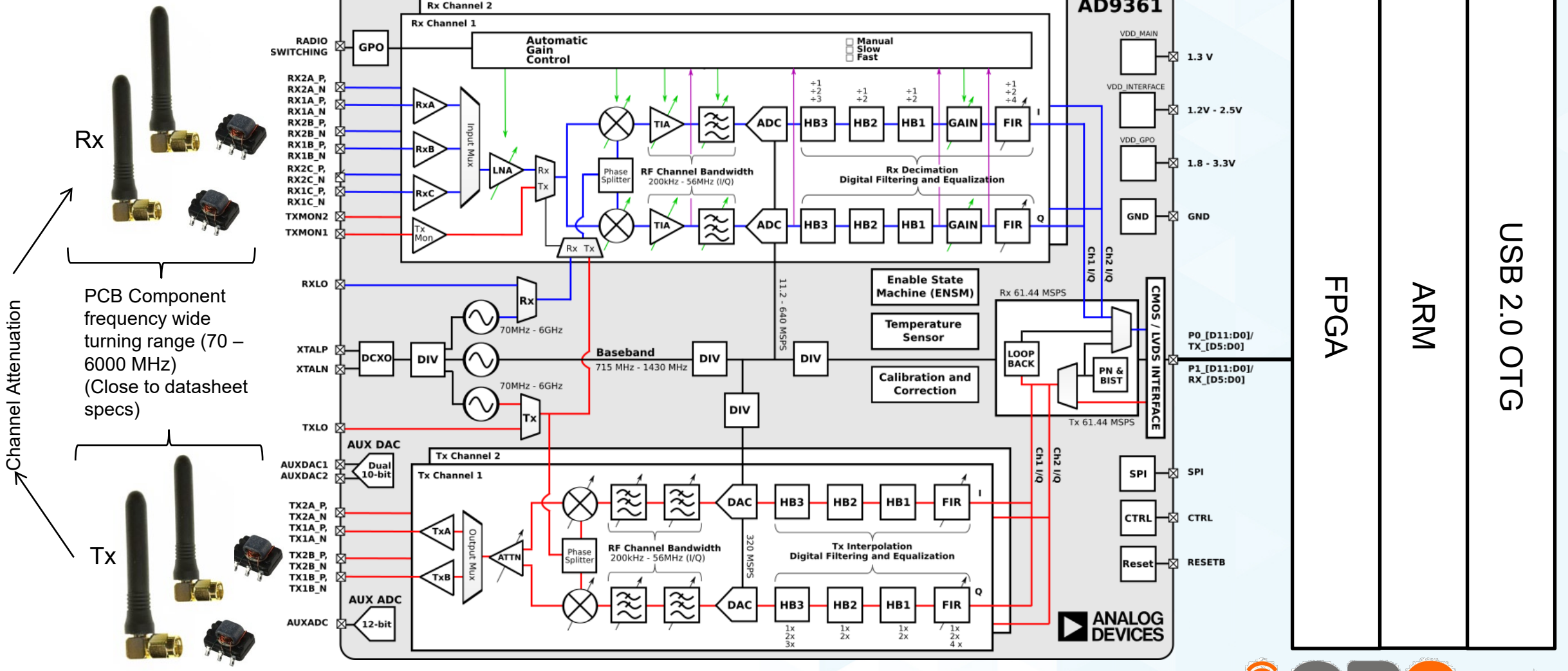
ADRV-DPD1
- *AD9375 + 250 mW PA*
- 2 Rx, 2 Rx
- LTE Band 7
  - 2500 to 2570 Uplink
  - 2620 to 2690 MHz Downlink
- 2 PAs, 2 LNAs, duplex filters

ADRV9009-ZU11EG
- *2 x ADRV9009 + Zynq Uiltrascale*
- 75MHz to 6GHz tuning range
- Rx BW 200MHz
- Tx synthesis bandwidth 450 MHz
- Integrated LO and Phase synch between all channels and Modules
- 4G x64 w/ECC PS; 4G (2Gb x32 x2Banks) PL
- USB3, USB2, PCIe 3.0 x8, QSFP+, SFP+, 1Gb Ethernet x2, and CPRI

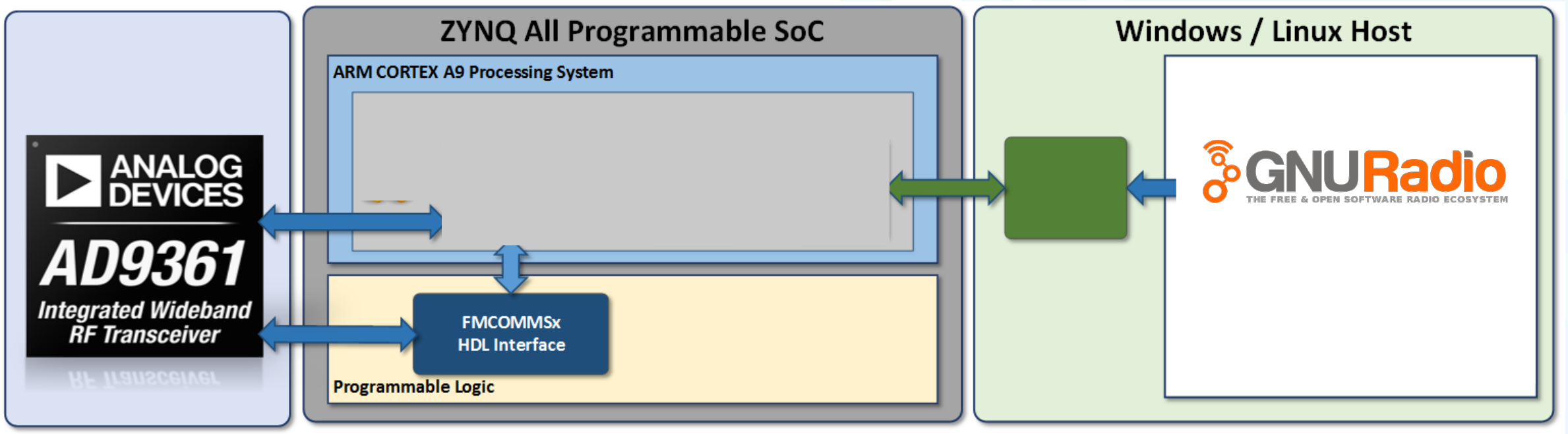
GRCon

Linux's Industrial Input/Output (IIO) infrastructure

ADI ZIF Transceivers

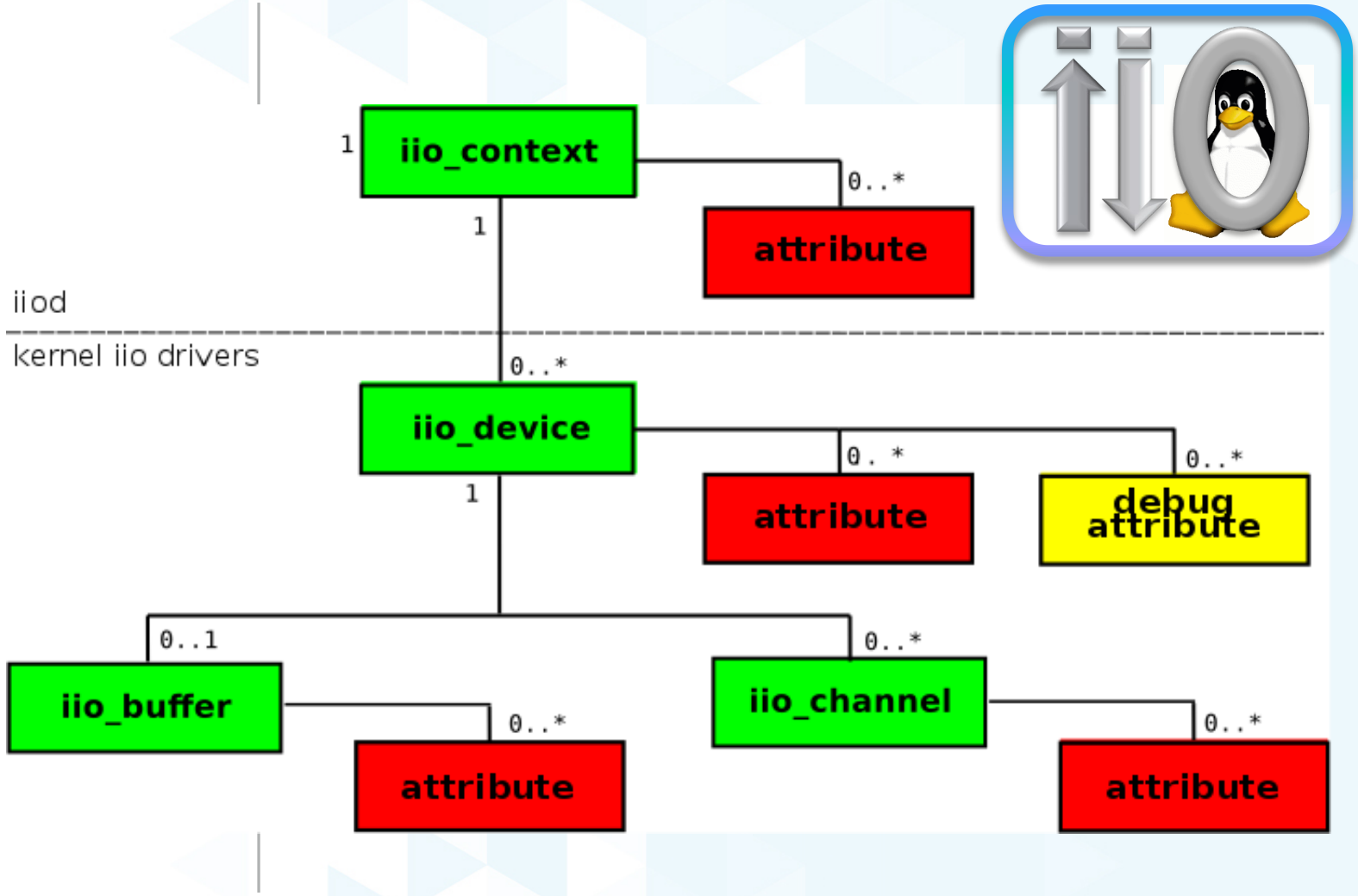


Radio to Host Interface

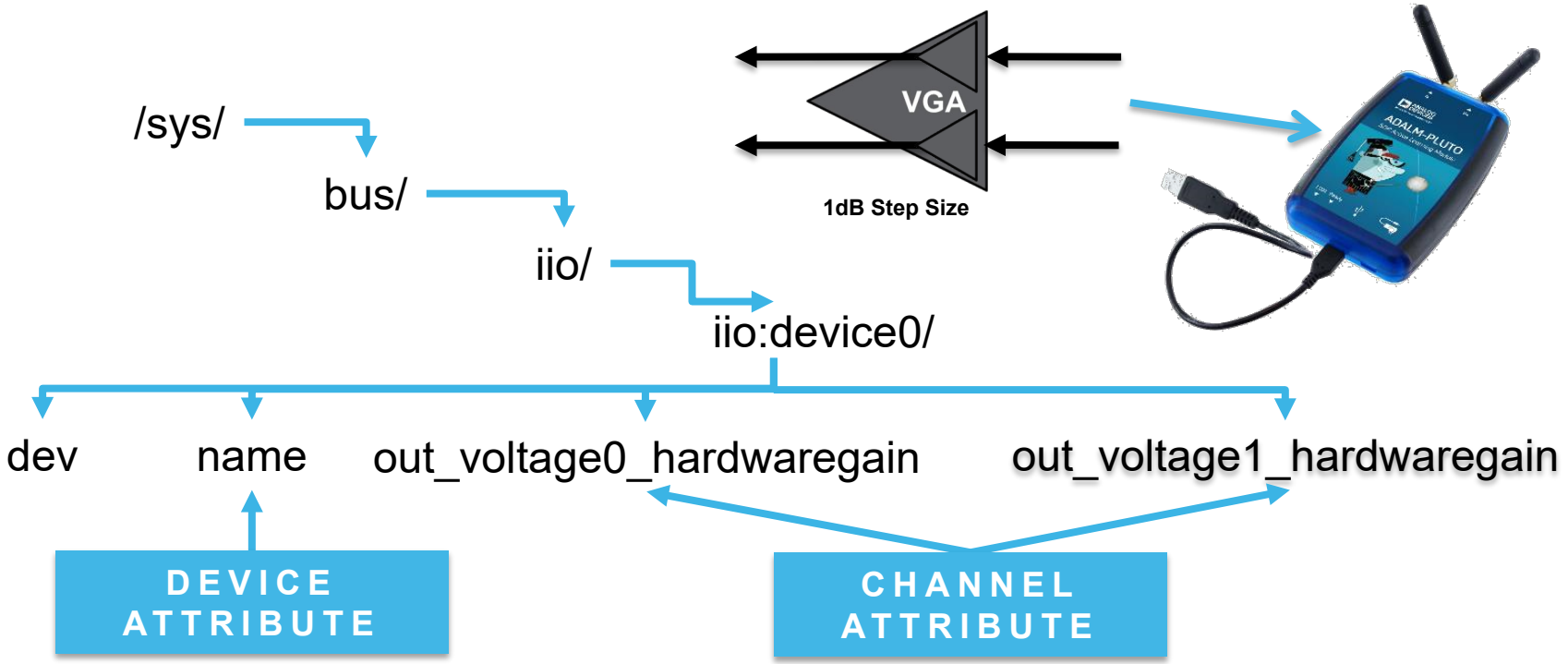


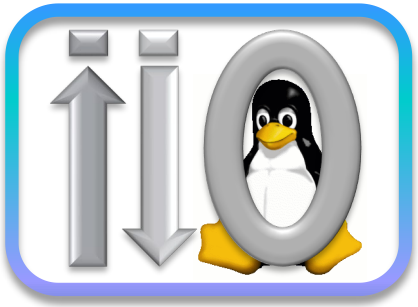
IIO Concepts

- The Linux Industrial I/O (IIO) subsystem is intended to provide support for devices that, in some sense, are analog-to-digital or digital-to-analog converters
  - Devices that fall into this category are:
    - ADCs
    - DACs
    - Accelerometers, gyros, IMUs
    - Capacitance-to-Digital converters (CDCs)
    - Pressure, temperature, and light sensors, etc.
    - RF Transceivers (like the AD9361 / AD9364 / AD9371 / ADRV9009)
  - Can be used on ADCs ranging from a 1MSPS SoC ADC to >5 GSPS ADCs



Pluto Gain Control

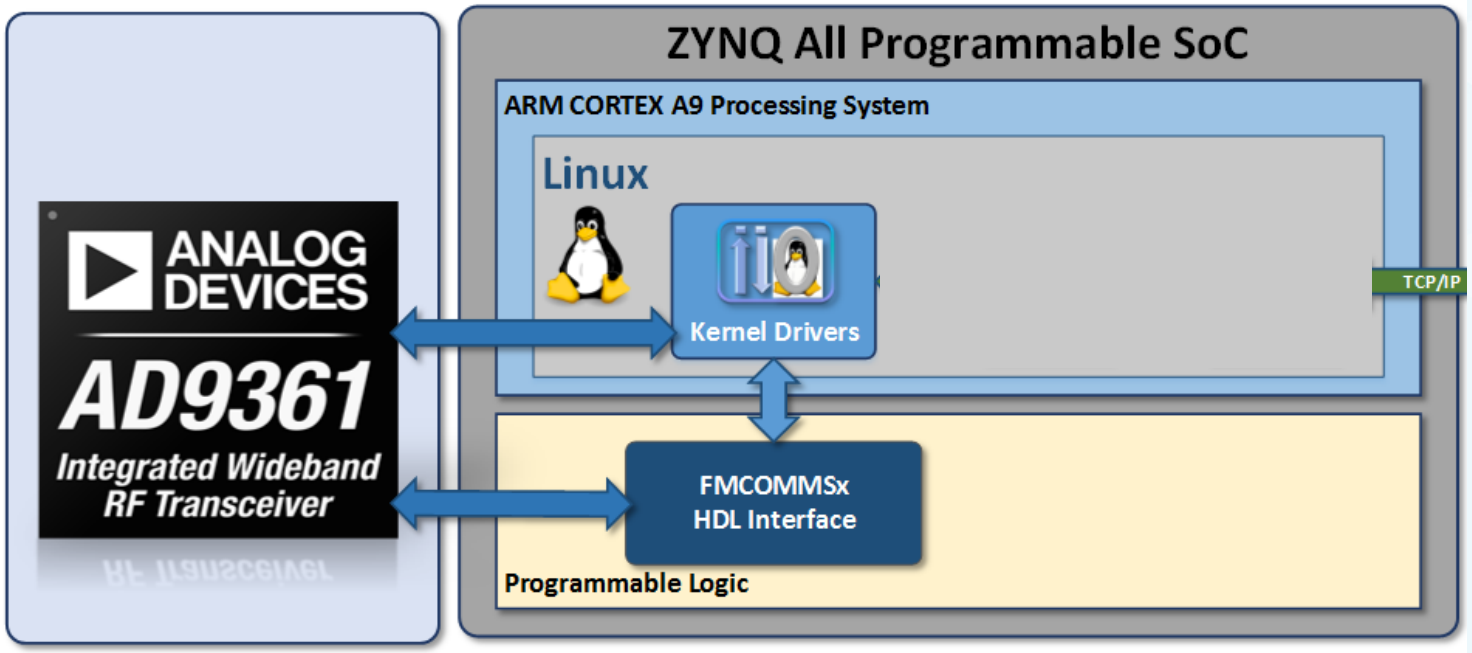




Shell Commands:

```
/sys/bus/iio/iio:device0 # cat name
ad8366-lpc
/sys/bus/iio/iio:device0 # echo 6 > out_voltage1_hardwaregain
/sys/bus/iio/iio:device0 # cat out_voltage1_hardwaregain
5.765000 dB
```

IIO Driver



Goal: How to I control the device?

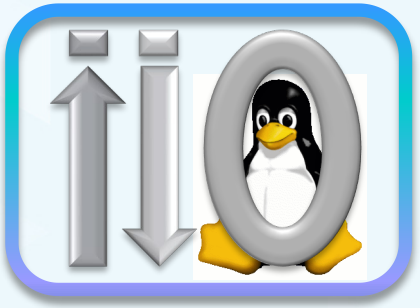
LO Frequency

Sample Rate

Gain Mode

TDD ENSM

RSSI



192.168.1.199 - PuTTY

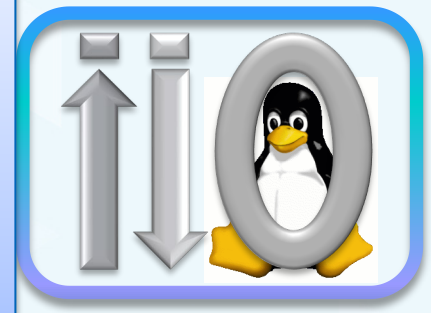
```
# ls
calib_mode                          in_voltage_rf_bandwidth                 out_voltage0_hardwaregain
calib_mode_available                in_voltage_rf_bandwidth_available       out_voltage0_hardwaregain_available
dcxo_tune_coarse                    in_voltage_rf_dc_offset_tracking_en     out_voltage0_rf_port_select
dcxo_tune_coarse_available          in_voltage_rf_port_select_available     out_voltage0_rssi
dcxo_tune_fine                      in_voltage_sampling_frequency           out_voltage2_raw
dcxo_tune_fine_available            in_voltage_sampling_frequency_available out_voltage2_scale
dev                                 multichip_sync                          out_voltage3_raw
ensm_mode                           name                                    out_voltage3_scale
ensm_mode_available                 of_node                                 out_voltage_filter_fir_en
filter_fir_config                   out_altvoltage0_RX_LO_external          out_voltage_rf_bandwidth
gain_table_config                   out_altvoltage0_RX_LO_fastlock_load     out_voltage_rf_bandwidth_available
in_out_voltage_filter_fir_en        out_altvoltage0_RX_LO_fastlock_recall   out_voltage_rf_port_select_available
in_temp0_input                      out_altvoltage0_RX_LO_fastlock_save     out_voltage_sampling_frequency
in_voltage0_gain_control_mode       out_altvoltage0_RX_LO_fastlock_store    out_voltage_sampling_frequency_available
in_voltage0_hardwaregain            out_altvoltage0_RX_LO_frequency         power
in_voltage0_hardwaregain_available  out_altvoltage0_RX_LO_frequency_available rssi_gain_step_error
in_voltage0_rf_port_select          out_altvoltage0_RX_LO_powerdown         rx_path_rates
in_voltage0_rssi                    out_altvoltage1_TX_LO_external          subsystem
in_voltage2_offset                  out_altvoltage1_TX_LO_fastlock_load     trx_rate_governor
in_voltage2_raw                     out_altvoltage1_TX_LO_fastlock_recall   trx_rate_governor_available
in_voltage2_scale                   out_altvoltage1_TX_LO_fastlock_save     tx_path_rates
in_voltage_bb_dc_offset_tracking_en out_altvoltage1_TX_LO_fastlock_store    uevent
in_voltage_filter_fir_en            out_altvoltage1_TX_LO_frequency         xo_correction
in_voltage_gain_control_mode_available out_altvoltage1_TX_LO_frequency_available xo_correction_available
in_voltage_quadrature_tracking_en   out_altvoltage1_TX_LO_powerdown
#
```

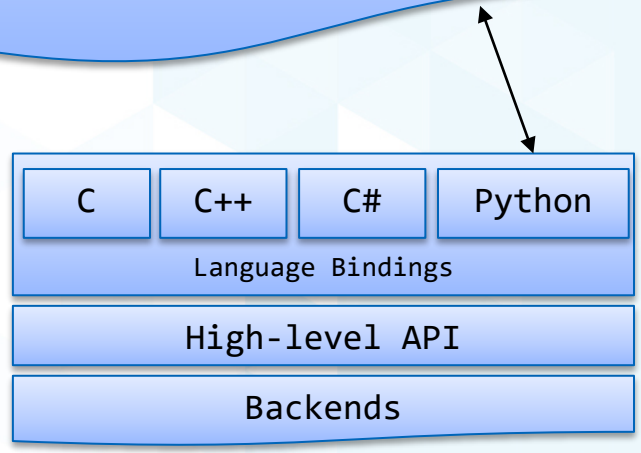
IIO – libiio

- System library
- Abstracts away low level details of the IIO kernel ABI
  - Kernel ABI is designed to be simple and efficient
  - libiio focuses on ease of use
- Provides high-level C, C++, C# or Python programming interface to IIO (Language bindings)

```
ctx = iio_create_context_from_uri("ip:192.168.2.1");

phy = iio_context_find_device(ctx, "ad9361-phy");

iio_channel_attr_write_longlong(
        iio_device_find_channel(phy, "altvoltage0", true),
        "frequency",
        2400000000); /* RX LO frequency 2.4GHz */

iio_channel_attr_write_longlong(
        iio_device_find_channel(phy, "voltage0", false),
        "sampling_frequency",
        5000000); /* RX baseband rate 5 MSPS */
```

```
ctx = iio.Context('ip:192.168.2.1')
phy = ctx.find_device('ad9361-phy')
chan1 = phy.find_channel('altvoltage0')

chan1.attrs['frequency'] = 24000000000

chan2 = phy.find_channel('voltage0')
chan2.attrs['sampling_frequency'] = 5000000
```





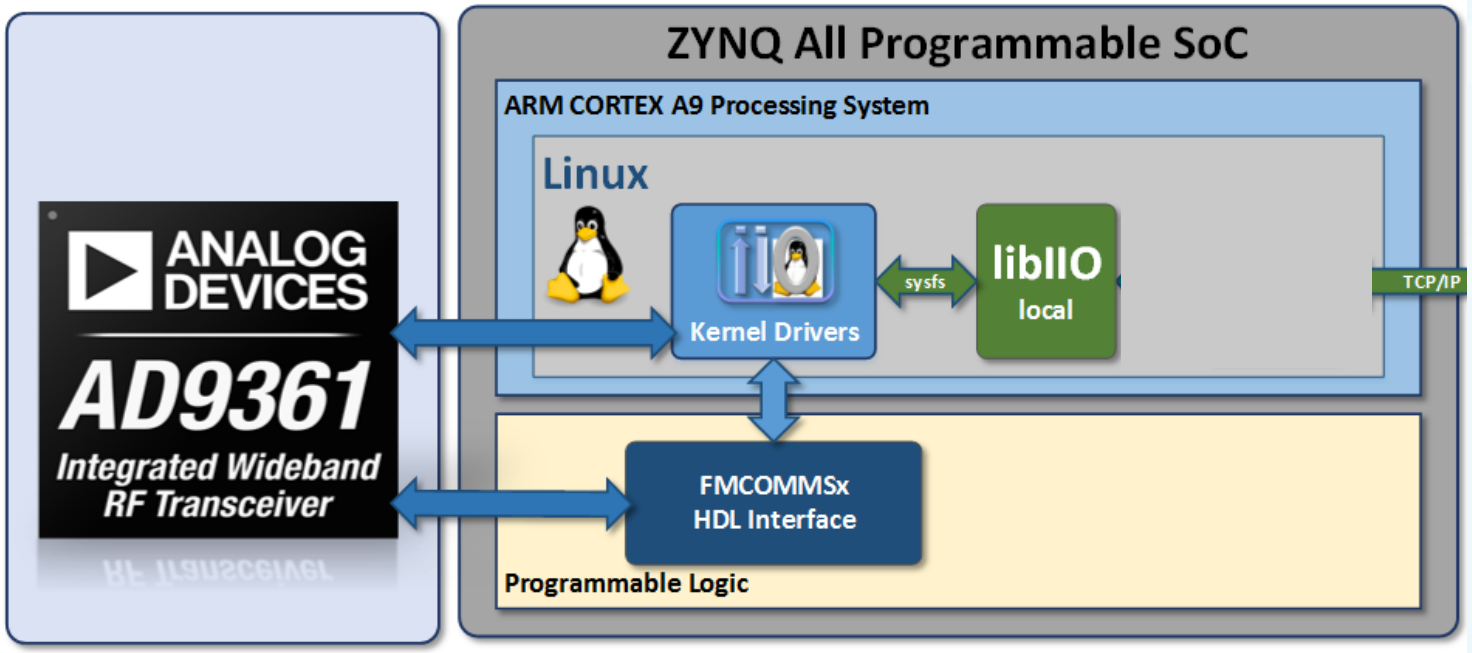
For more information:

https://github.com/analogdevicesinc/libiio

http://wiki.analog.com/resources/tools-software/linux-software/libiio_internals

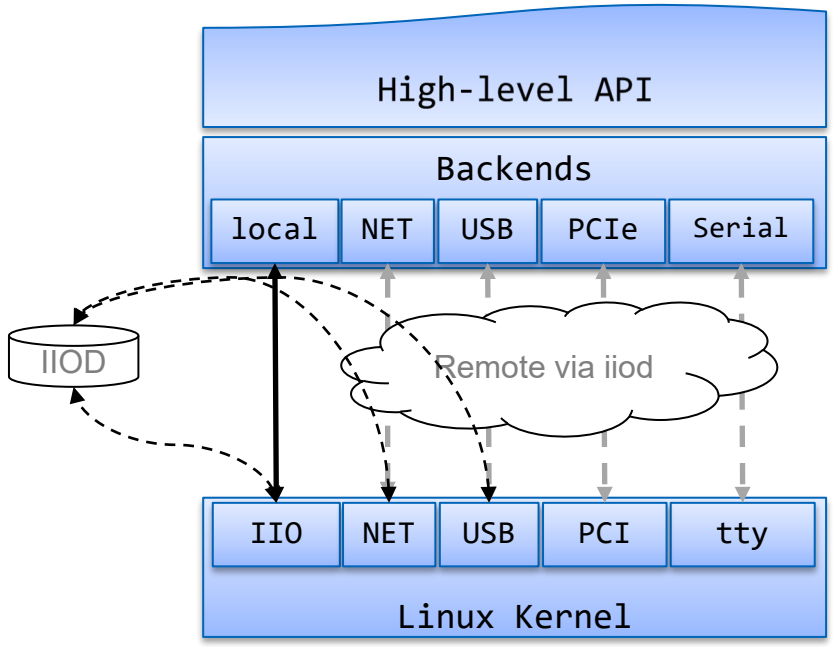
http://analogdevicesinc.github.io/libiio/

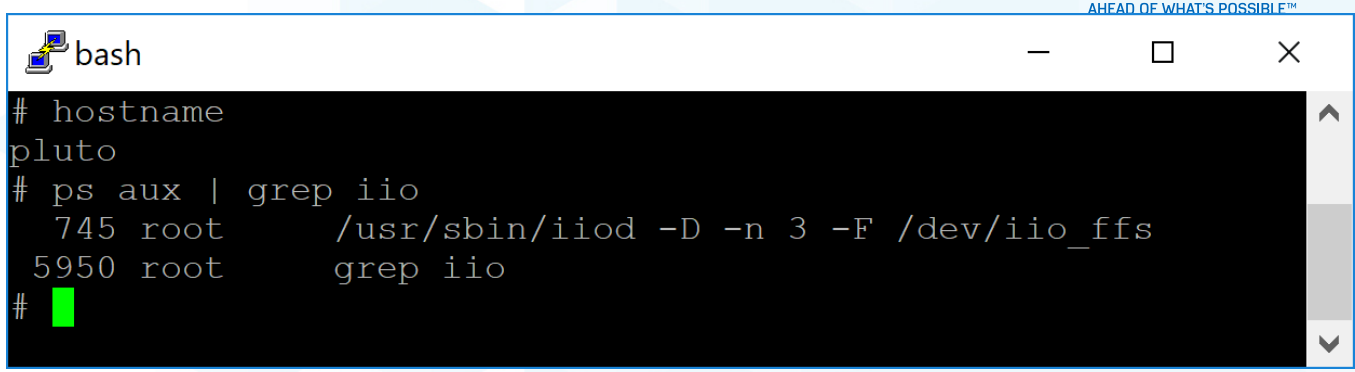
IIO Driver

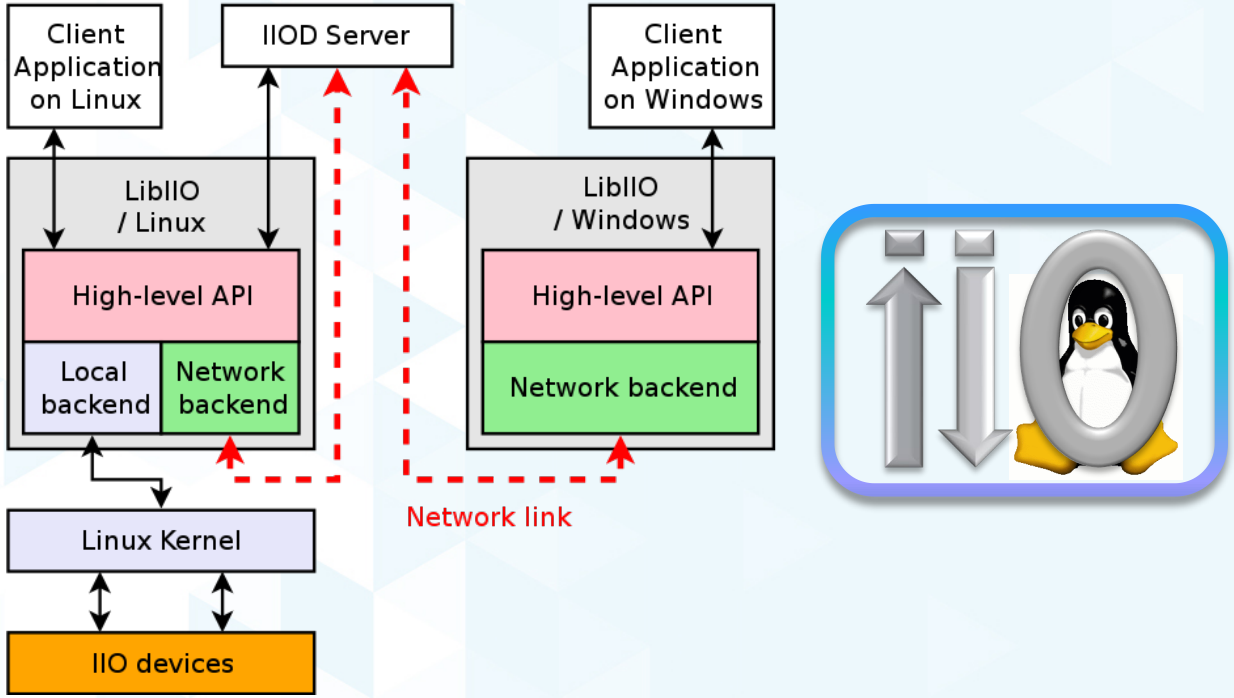


IIO – libiio – Backends

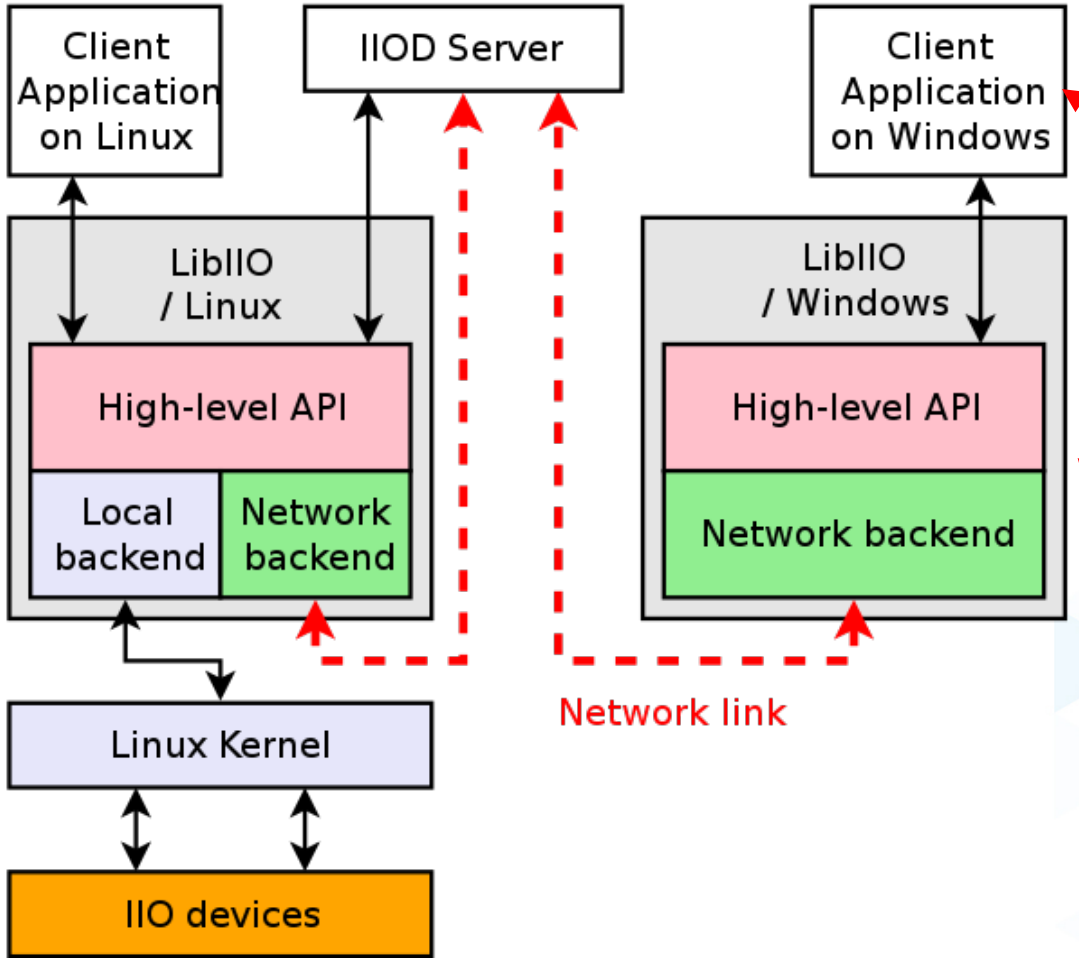
- Support for backends
  - Backend takes care of low-level communication details
  - Provide the same API for applications
  - Transparent from the applications point of view







libIIO and applications

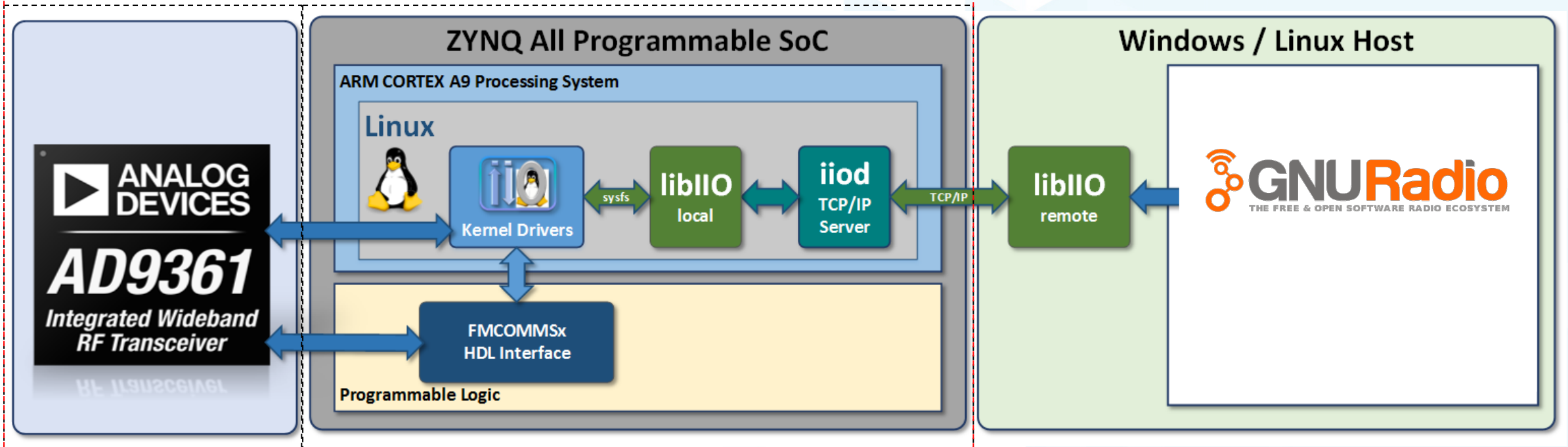


- *iio-utils (command line tools)*
- *IIO-Scope*
- *MATLAB and Simulink*
- GNU Radio
- 3rd part apps
  - SDR Angel
  - GQRX
  - SDR#
  - …

- Language support for libIIO
  - C/C++
  - Python
  - Nodejs (3rd party)
  - C# (3rd part)

IIO Driver

ADALM-PLUTO,
ADRV9361-Z7035 RF SOM, ADRV9364-Z7020 RF SOM, ADRV9009-ZU11EG RF SOM

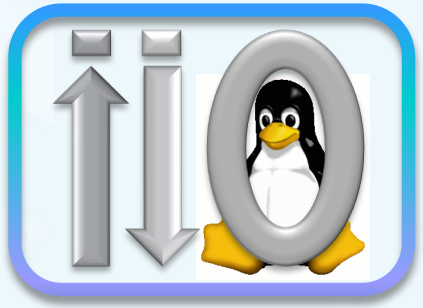


FMCOMMS2, FMCOMMS3,
FMCOMMS4, FMCOMMS5,
ADRV9371 ADRV9375
ADRV9009

Xilinx Zed Board, Xilinx ZC702, Xilinx ZC706,
Xilinx KC705, Xilinx KCU105,
Xilinx ZCU102
Intel Arria 10 SoC, Intel Stratix 10 SoC

IIO – libiio – Command line tools

- **iio_info** : Information about all IIO devices, backends and context attributes
  - `iio_info -s`
  - `iio_info -u ip:192.168.2.1`



- **iio_attr** : Read and write IIO attributes
  - `iio_attr -c ad9361-phy altvoltage0 frequency 2450000000`

- **iio_readdev** : Read samples from an IIO device
  - `iio_readdev -u usb:1.100.5 -b 100000 cf-ad9361-lpc | pv > /dev/null`

- **iio_writedev** : Write samples to an IIO device
  - `iio_readdev -b 100000 cf-ad9361-lpc | iio_writedev -b 100000 cf-ad9361-dds-core-lpc`

- **iio_reg** : Read or write SPI or I2C registers in an IIO device (useful to debug drivers)
  - `iio_reg adrv9009-phy 0`

Discovery & Resolution

(Linux, Windows or macOS Host, Using iio-utils)

ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

- Are there any iio devices on the local domain?

```
rgetz@brain:~/github/libiio$ iio_info -S
Library version: 0.21 (git tag: 565bf68)
Compiled with backends: local xml ip usb serial
Available contexts:
    0: 0456:b673 (Analog Devices Inc. PlutoSDR (ADALM-PLUTO)), serial=1044739659930006feff0f0010e1273635 [usb:3.33.5]
    1: 192.168.1.116 (AD-FMCOMMS2-EBZ on Xilinx Zynq ZED (armv7l)), serial=00100 [ip:analog-2.local]
    2: 192.168.2.1 (Analog Devices PlutoSDR Rev.C (Z7010-AD9363A)), serial=1044739659930006feff0f0010e1273635 [ip:pluto.local]
```

- Are there any iio devices on the local domain?

```
rgetz@brain:~/github/libiio$ iio_attr -S
Available contexts:
    0: 0456:b673 (Analog Devices Inc. PlutoSDR (ADALM-PLUTO)), serial=1044739659930006feff0f0010e1273635 [usb:3.33.5]
    1: 192.168.2.1 (Analog Devices PlutoSDR Rev.C (Z7010-AD9363A)), serial=1044739659930006feff0f0010e1273635 [ip:pluto.local]
    2: 192.168.1.116 (AD-FMCOMMS2-EBZ on Xilinx Zynq ZED (armv7l)), serial=00100 [ip:analog-2.local]
```

- Are there any iio devices on the local domain? (specifically ethernet)

```
rgetz@brain:~/github/libiio$ iio_attr -S ip
Available contexts:
    0: 192.168.2.1 (Analog Devices PlutoSDR Rev.C (Z7010-AD9363A)), serial=1044739659930006feff0f0010e1273635 [ip:pluto.local]
    1: 192.168.1.116 (AD-FMCOMMS2-EBZ on Xilinx Zynq ZED (armv7l)), serial=00100 [ip:analog-2.local]
```

pyadi-iio

ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

- IIO version

```
32  # Setup contexts
33  try:
34      ctx = iio.Context('ip:192.168.2.1')
35  except:
36      print("No device found")
37      sys.exit(0)
38
39  ctrl = ctx.find_device("ad9361-phy")
40  txdac = ctx.find_device("cf-ad9361-dds-core-lpc")
41  rxadc = ctx.find_device("cf-ad9361-lpc")
42
43  # Configure transceiver settings
44  rxLO = ctrl.find_channel("altvoltage0", True)
45  rxLO.attrs["frequency"].value = str(int(RXLO))
46  txLO = ctrl.find_channel("altvoltage1", True)
47  txLO.attrs["frequency"].value = str(int(TXLO))
48
49  tx = ctrl.find_channel("voltage0",True)
50  tx.attrs["rf_bandwidth"].value = str(int(RXBW))
51  tx.attrs["sampling_frequency"].value = str(int(RXFS))
52  tx.attrs['hardwaregain'].value = '-30'
53
54  rx = ctrl.find_channel("voltage0")
55  rx.attrs["rf_bandwidth"].value = str(int(TXBW))
56  rx.attrs["sampling_frequency"].value = str(int(TXFS))
57  rx.attrs['gain_control_mode'].value = 'slow_attack'
58
```

- Pyadi-iio version

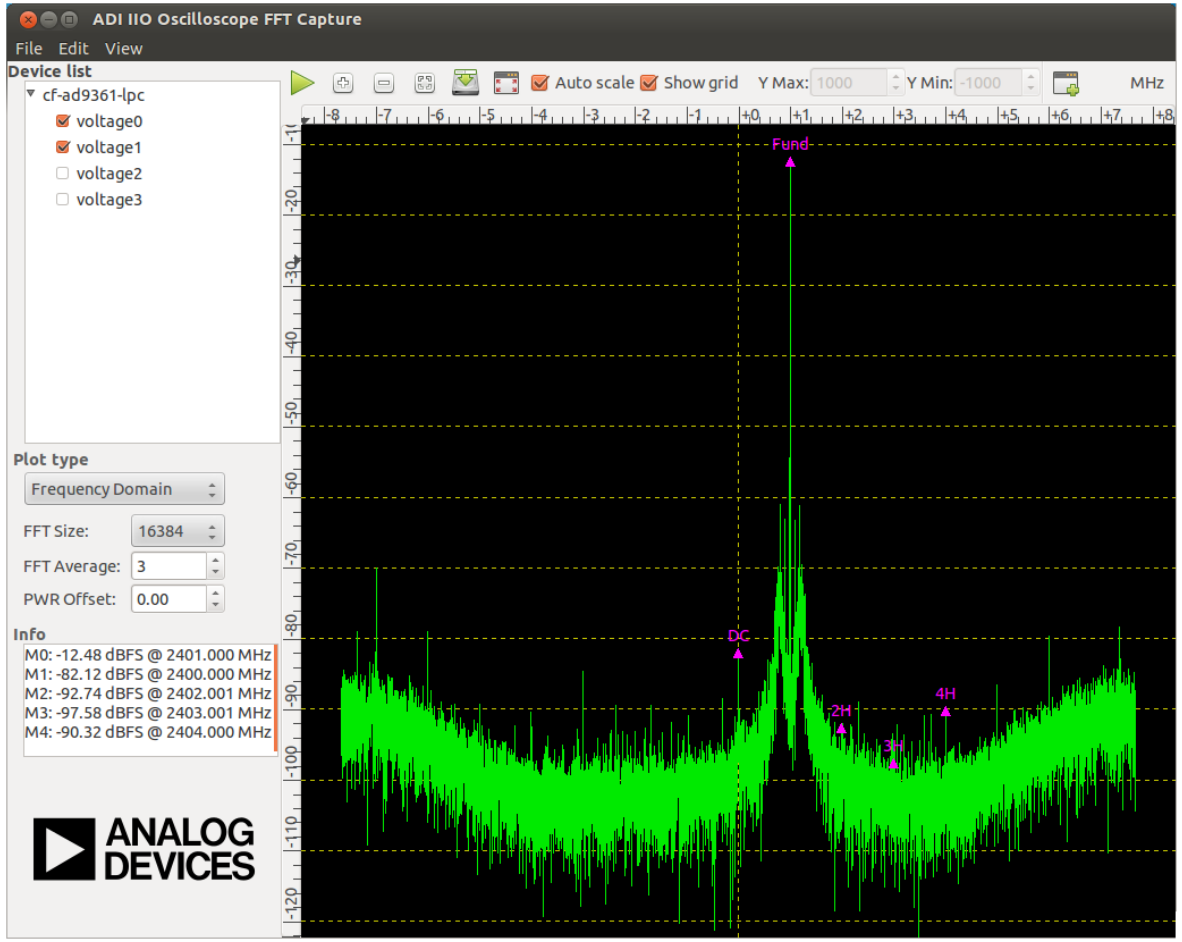
```
41  # Create radio
42  sdr = adi.Pluto()
43
44  # Configure properties
45  sdr.rx_rf_bandwidth = 4000000
46  sdr.rx_lo = 2000000000
47  sdr.tx_lo = 2000000000
48  sdr.tx_cyclic_buffer = True
49  sdr.tx_hardwaregain = -30
50  sdr.gain_control_mode = "slow_attack"
51
```

pip install pyadi-iio

GRCon

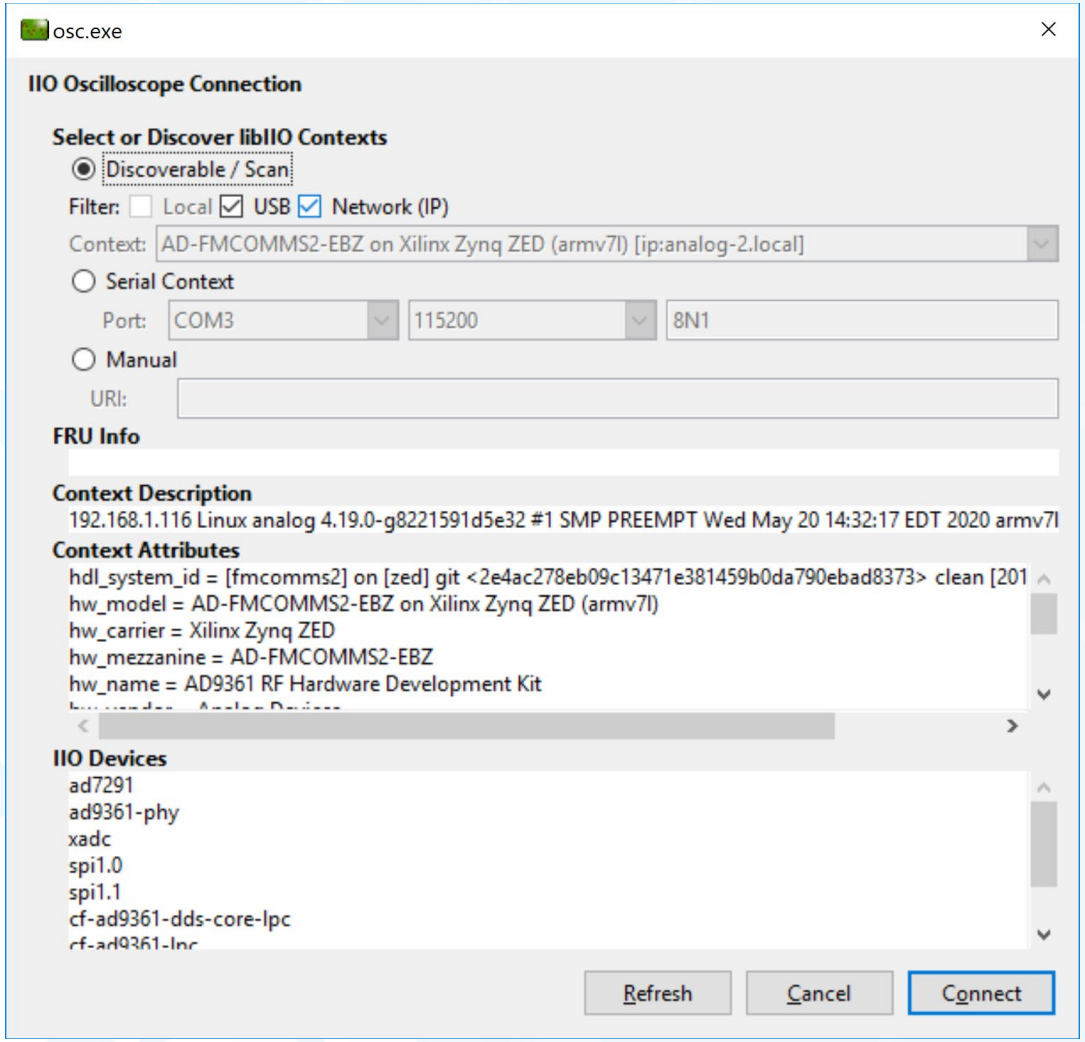
IIO-Scope

- Capture and display data
  - Time domain
  - Frequency domain
  - Constellation plot
  - Markers
  - Math operations
- Device configuration
- Plug-in system allow to create device or complex specialized GU
- Should support any IIO device
- Cross platform



IIO-Scope Zeroconf

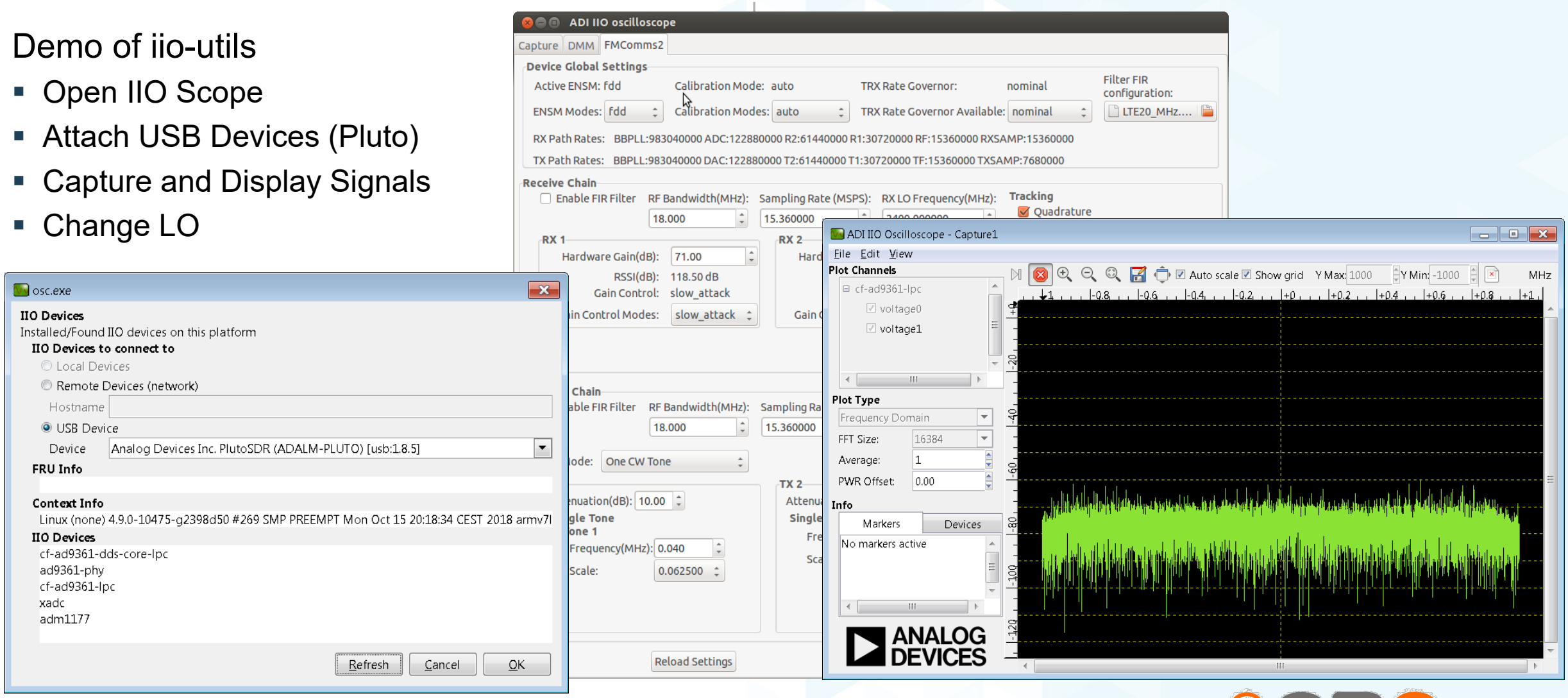
- GUI can use libiio functions, and get a list of any discoverable IIO devices:
  - USB
  - Network
  - local (Linux hosts only)
- Since GUI has access to both IP number, and context name, it keeps track and doesn't need Bonjour on Windows
  - It resolves the address itself.



IIO-Tools + IIO-Scope

Instructor-Led Demo

- Demo of iio-utils
  - Open IIO Scope
  - Attach USB Devices (Pluto)
  - Capture and Display Signals
  - Change LO

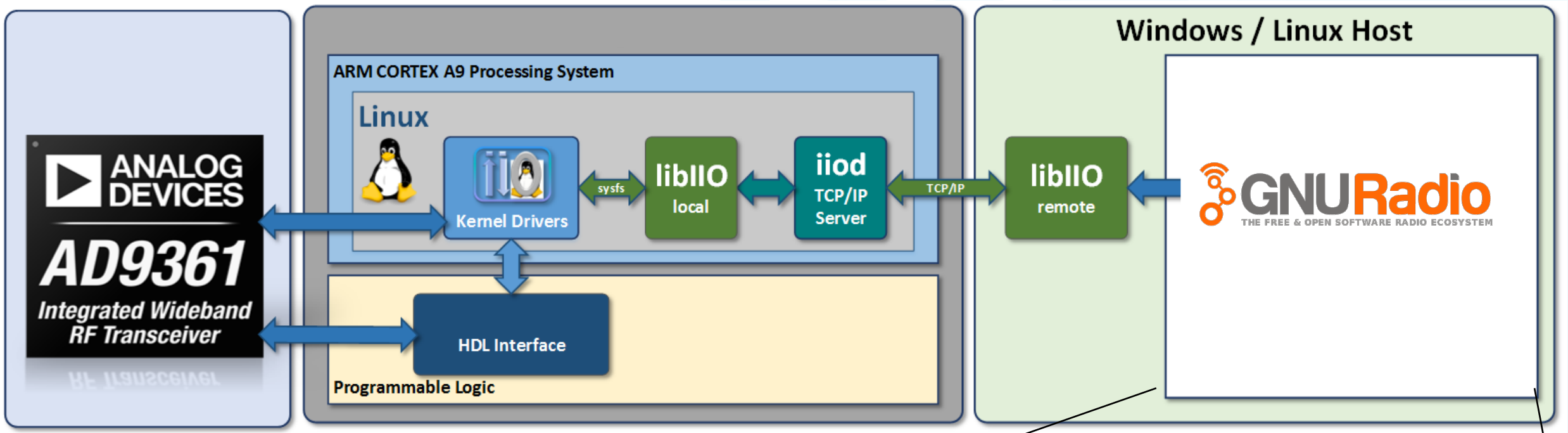


Questions about IIO-Tools + IIO-Scope?

- IIO scope is cross platform
  - Linux, MAC, Windows
  - i386, amd64, arm (on Zynq, or Raspberry Pi)
  - Runs remote or local
    - Local requires linux
    - Lots of local devices on standard laptops
- IIO has lots of on-line documentation
  - wiki.analog.com

http://www.iconarchive.com/show/noto-emoji-objects-icons-by-google/62807-radio-icon.html
http://www.streamlineicons.com
http://pixelkit.com

Why can't I use IIO Scope for Everything?



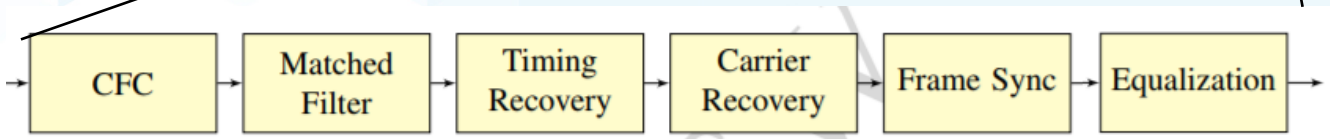
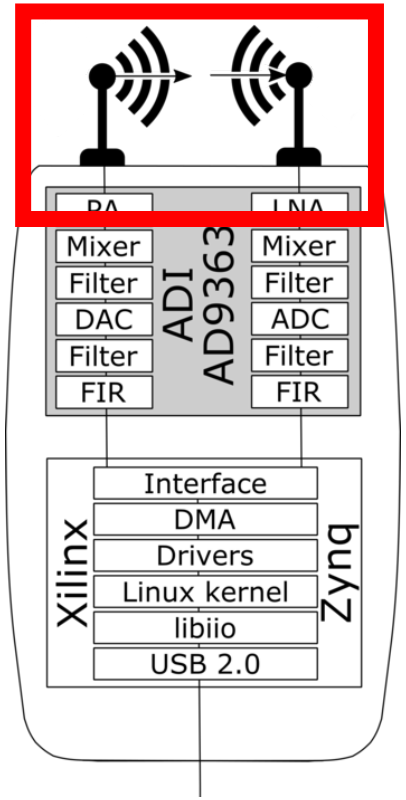


Figure 9.1: Receiver Block Diagram.

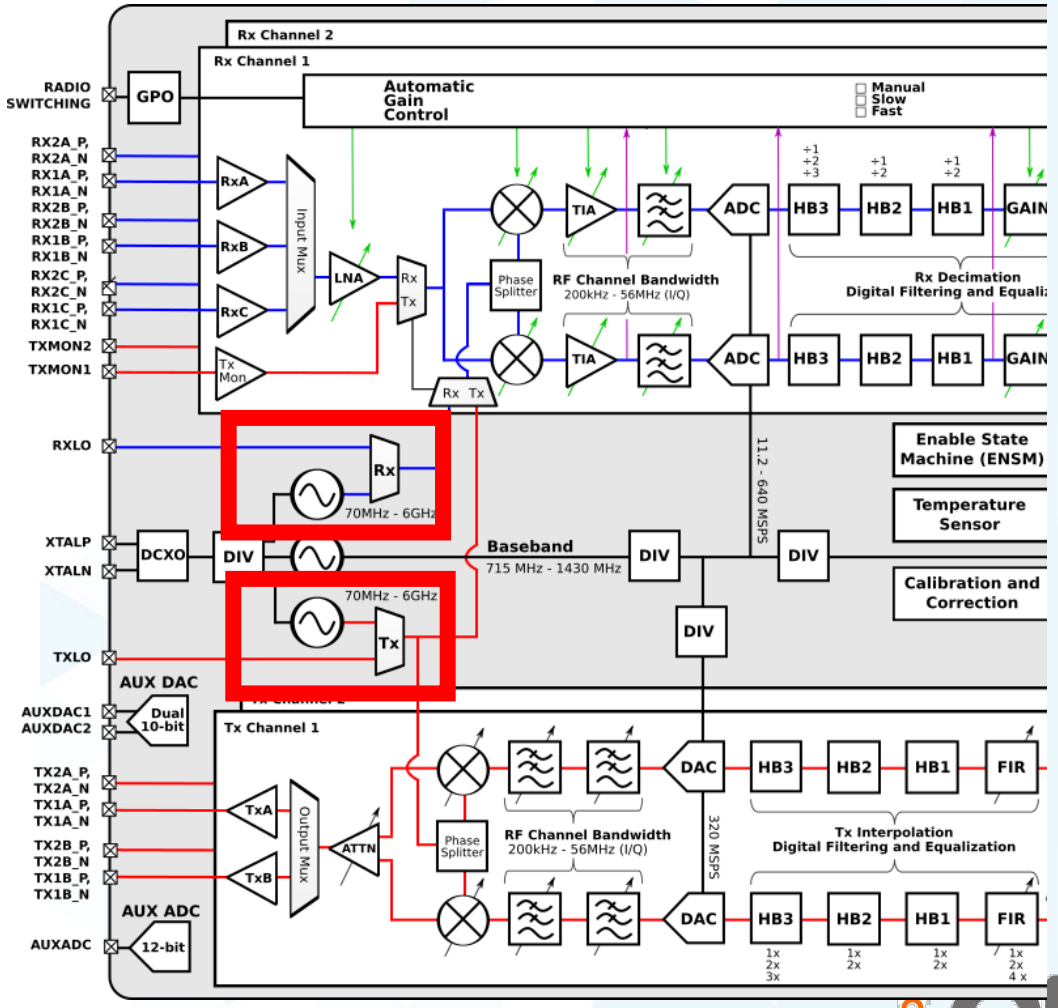
Understanding Sources of “Offset”

Analog Domain Propagation

Light travels ~33.6 ps/cm
1 wavelength @ 2 GHz = 500ps
1 cm = 24° phase shift @ 2 GHz



Separate PLLs

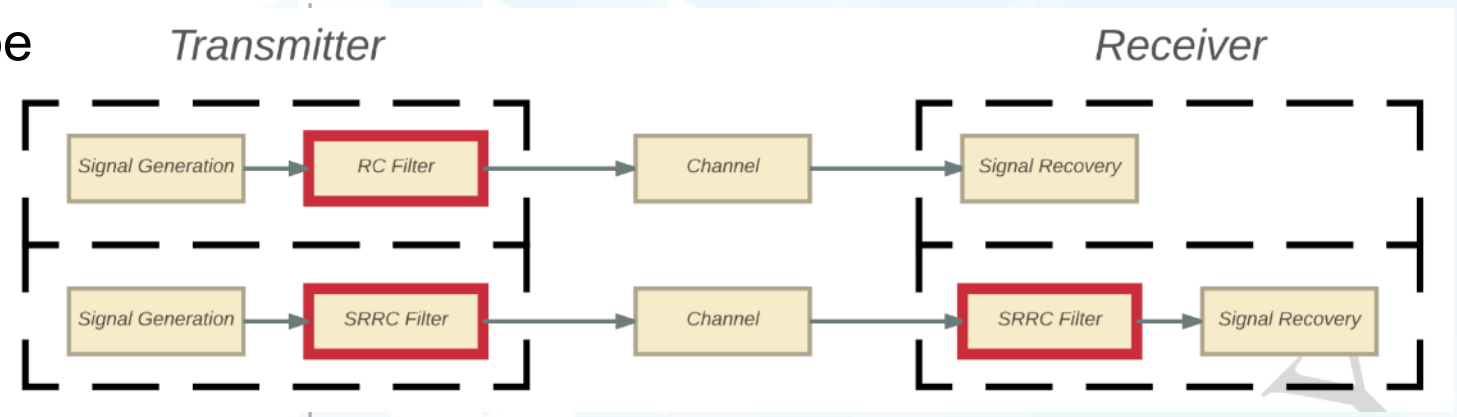


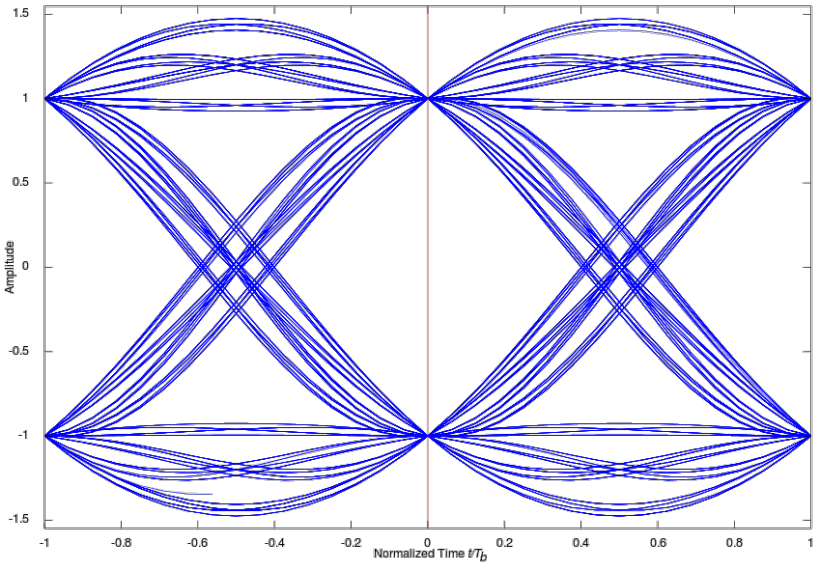
Timing Offset

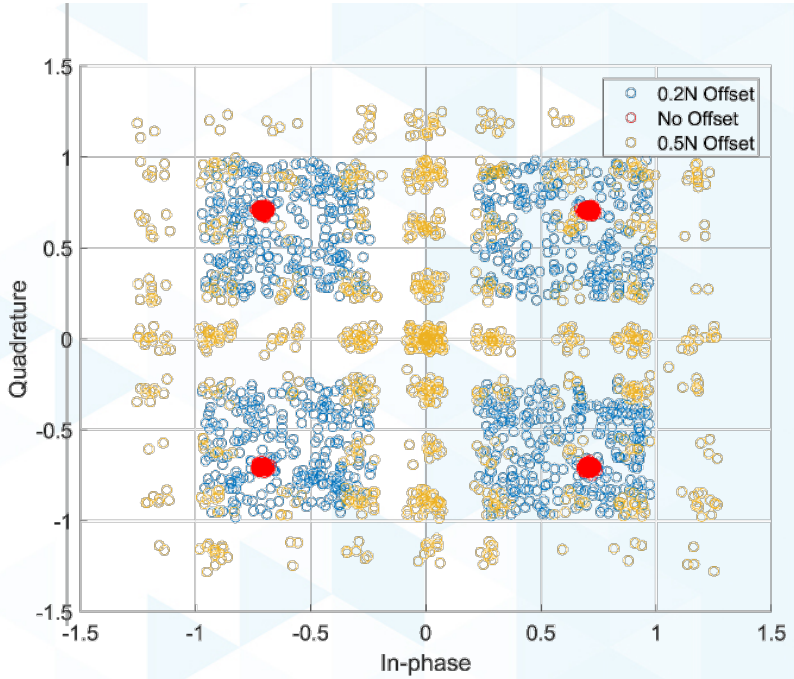
- Fractional and integer offsets can be modeled as:

$$r(t) = \sum_n x(n)h(t - \tau(t) - nT_s) + v(t),$$

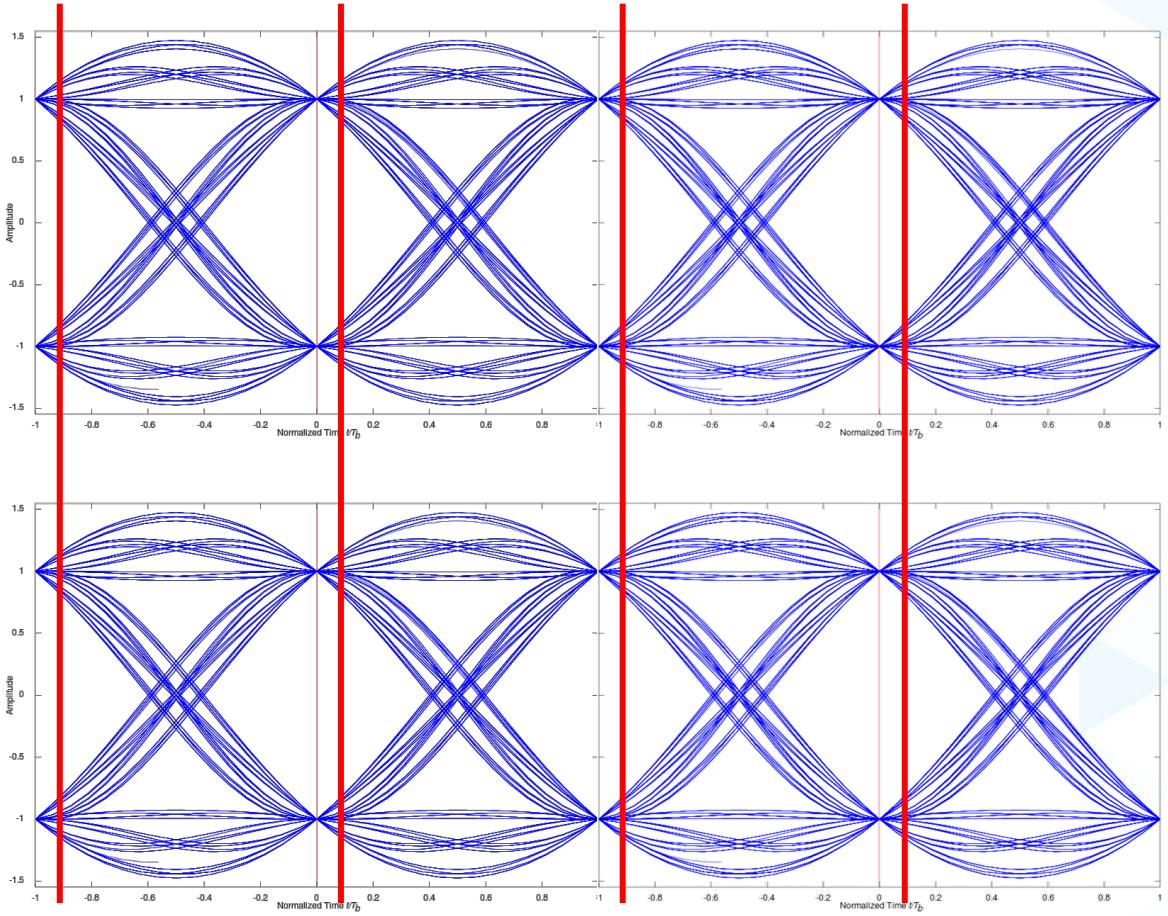
- Upsampling and selective decimation compensates for this problem

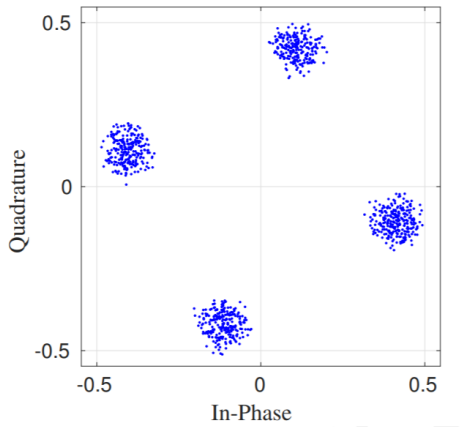




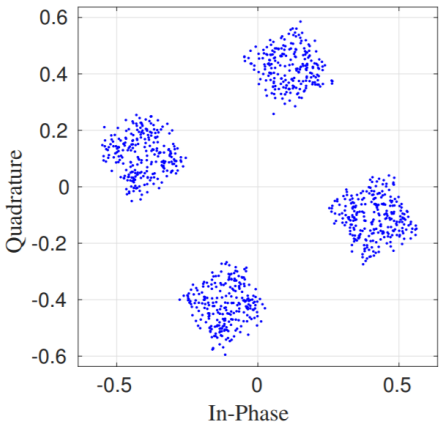


Timing Offset

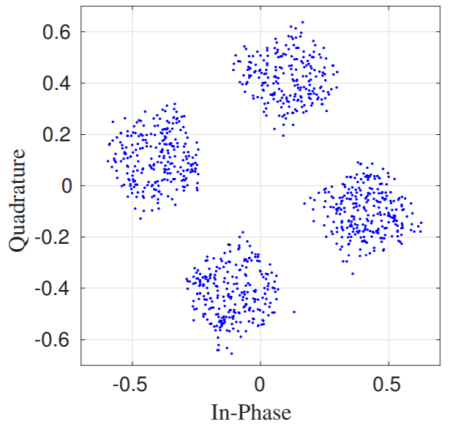




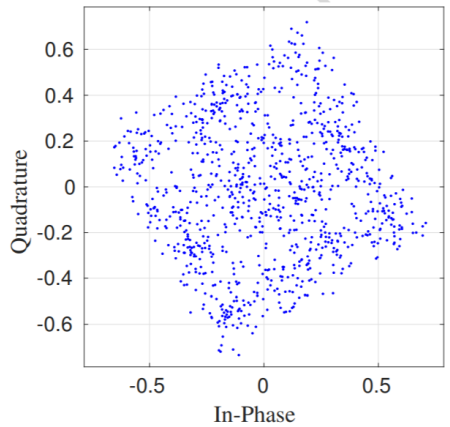
(b) $\hat{\tau} = 0.5$



(a) $\hat{\tau} = 0.1$

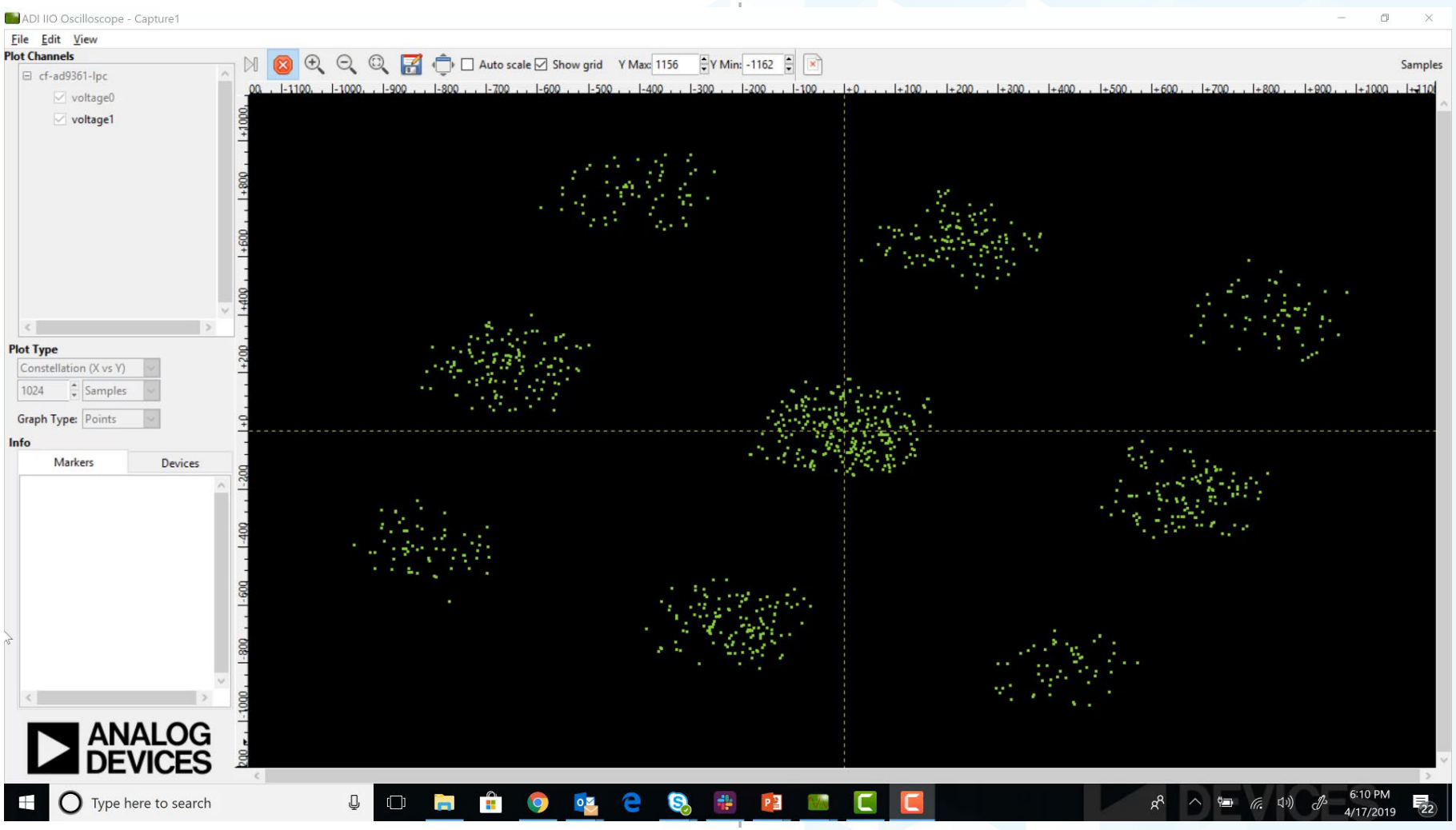


(c) $\hat{\tau} = 1$

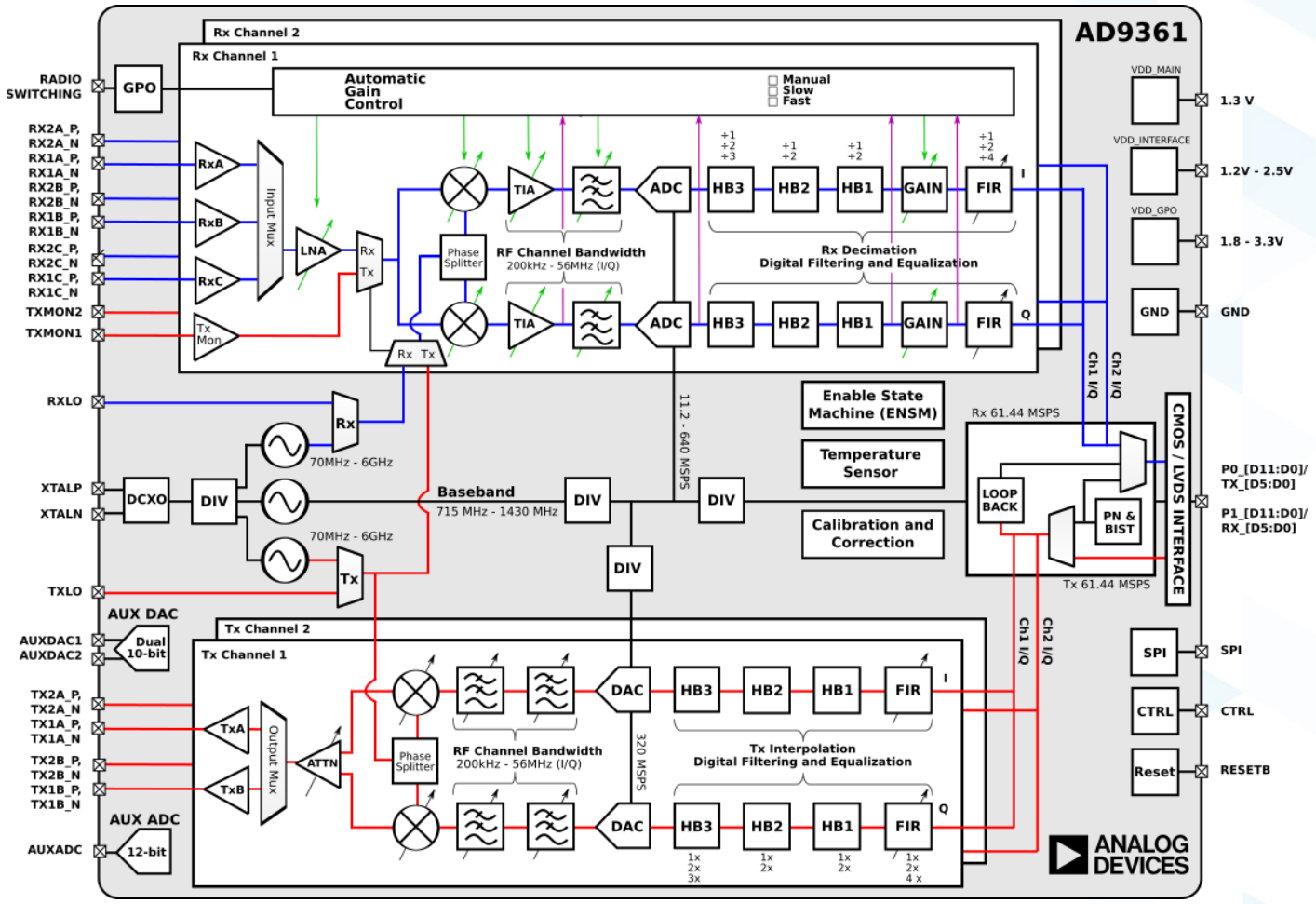


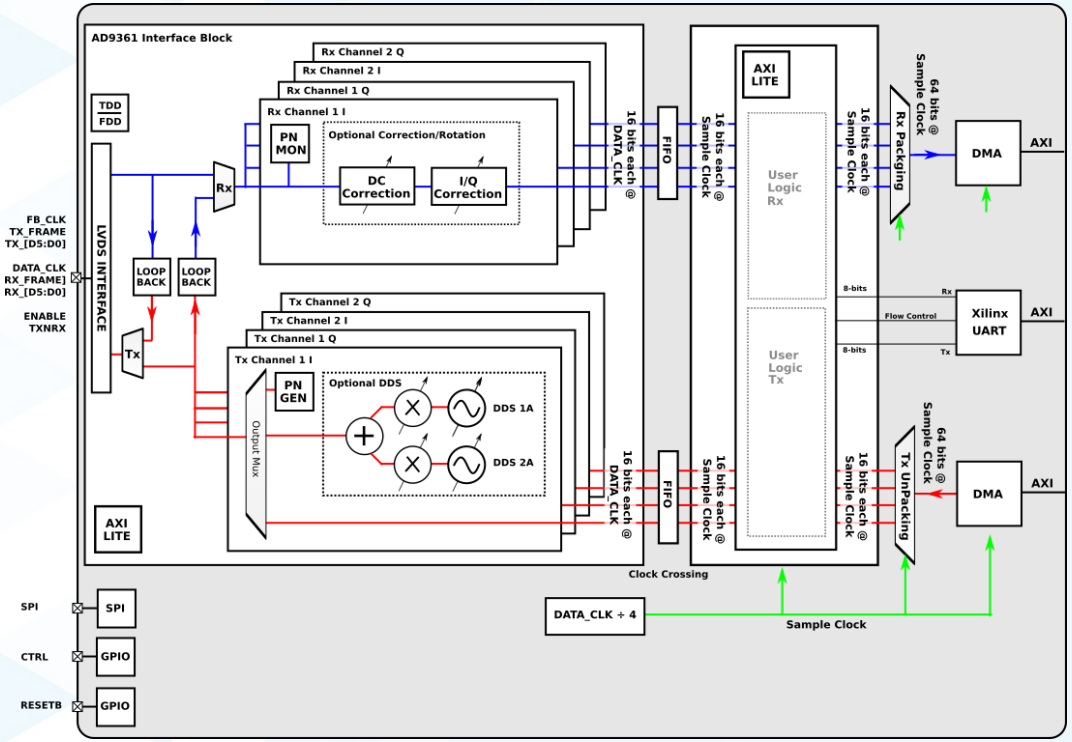
(d) $\hat{\tau} = 1.5$

Carrier Offset



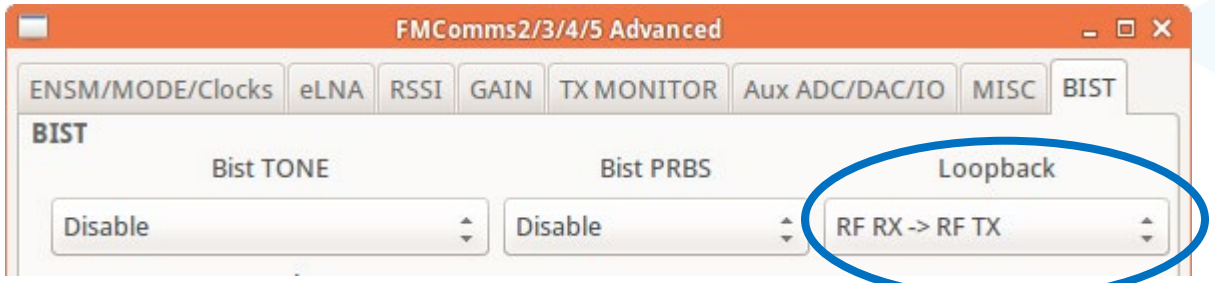
FPGA Interface

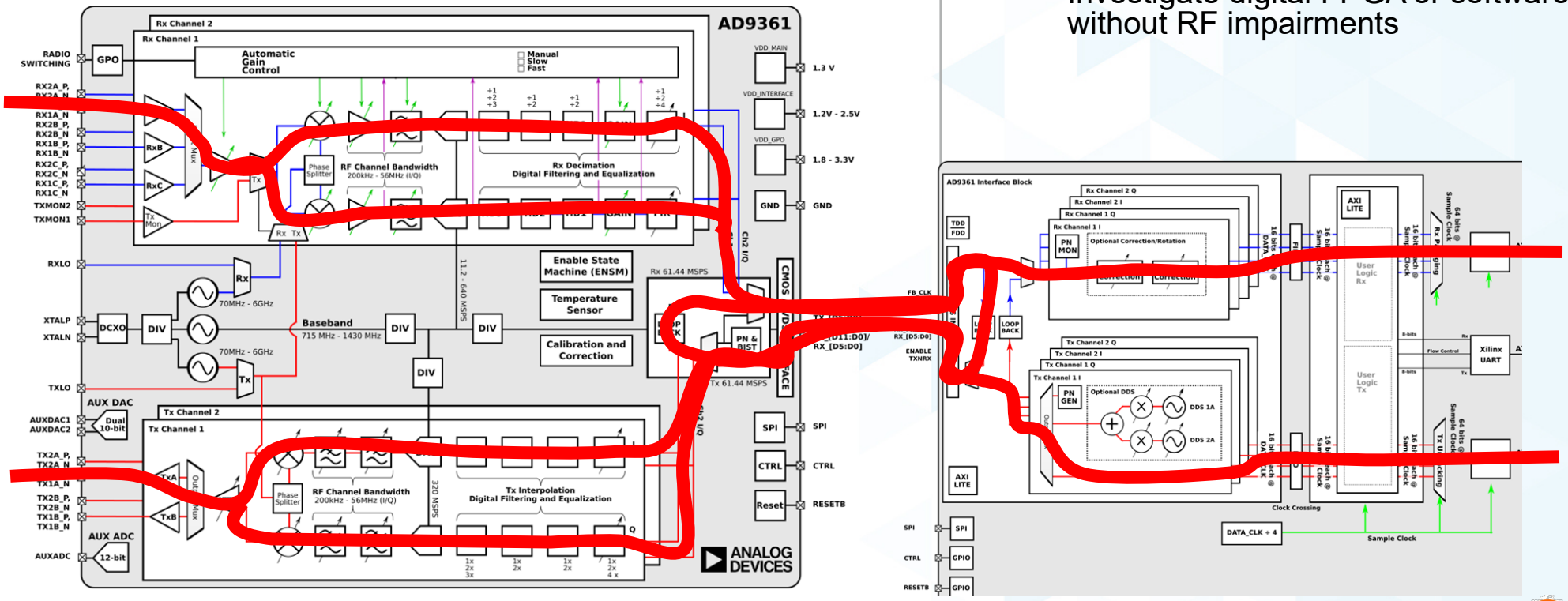




Loopback Paths

- Reuse the loopback paths for testing and development
  - Investigate hardware RF issues without software or FPGA interaction
    - Signal still digitalized, still goes through filters
    - will see RF impairments due to AD9361 setup
    - Output will be baseband copy of input
    - LO (Rx/Tx) interaction will occur if the same frequency – use different frequencies
  - Investigate digital FPGA or software algorithm issues without RF impairments





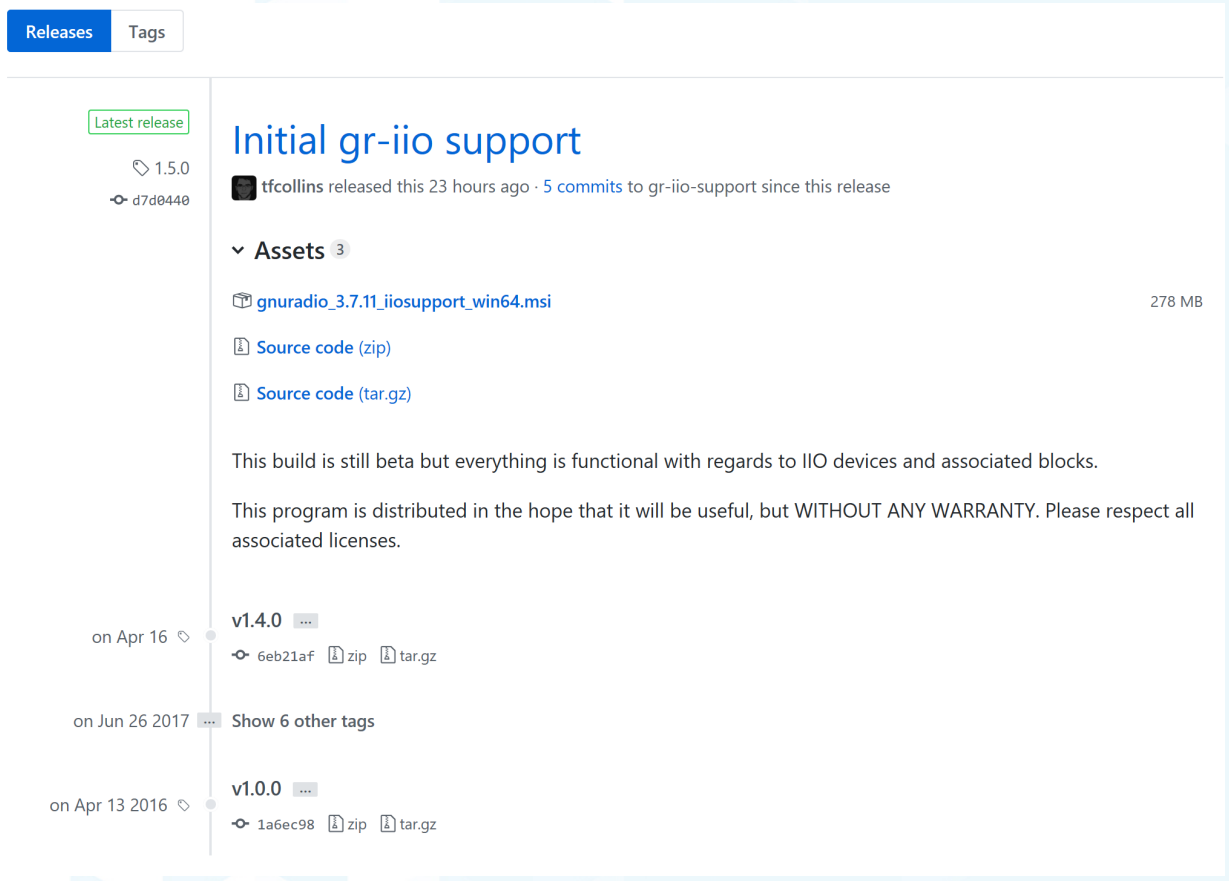
GNU Radio + iio = gr-iio

AHEAD OF WHAT'S POSSIBLE™

gr-iio: Details

- Repository
  - https://github.com/analogdevicesinc/gr-iio
  - **In gnuradio master since July!!!!**
- Dependencies
  - libiio
  - libad9361-iio

```
git clone https://github.com/analogdevicesinc/gr-iio.git
cd gr-iio
cmake .
make
sudo make install
cd ..
sudo ldconfig
```



IIO GNU Radio Support: gr-iio

- Industrial IO
  - FMComms
    - FMComms2/3/4 Sink
    - FMComms2/3/4 Source
    - FMComms5 Sink
    - FMComms5 Source
  - IIO Attribute Sink
  - IIO Attribute Source
  - IIO Attribute Updater
  - IIO Device Sink
  - IIO Device Source
  - Math Operators
    - Function
    - Modulo
    - Modulo Const
    - Power
  - PlutoSDR
    - PlutoSDR Sink
    - PlutoSDR Source
  - Waveform Generators
    - Function Generator

SDR

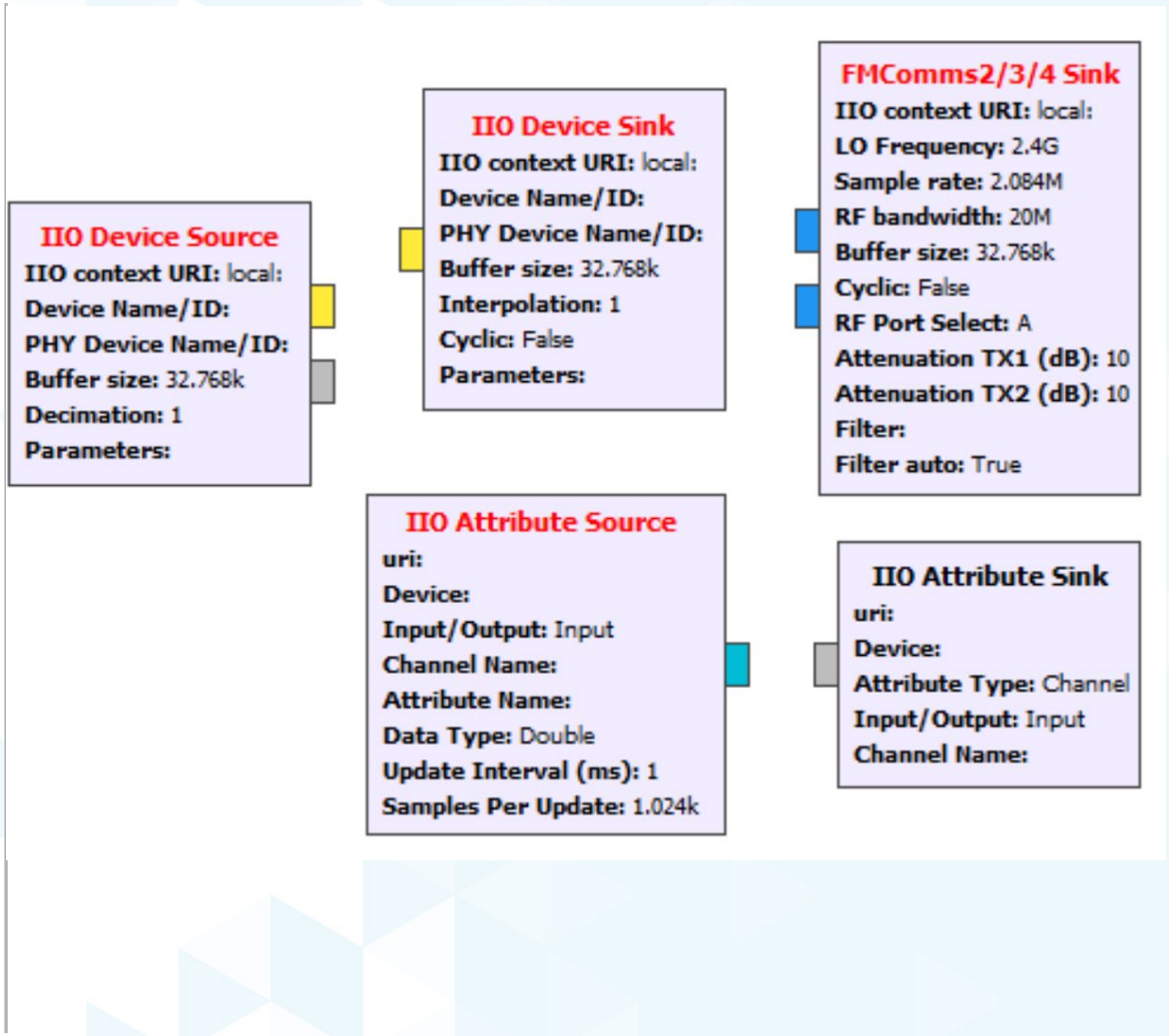
Attributes

Streams

Math (Scopy)

SDR

Math (Scopy)



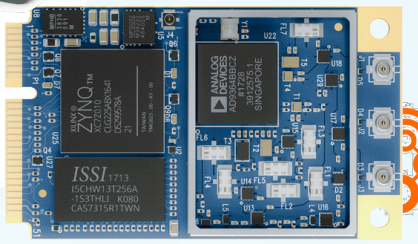
Hardware Support Through IIO

IIO Device Source
IIO context URI: local:
Device Name/ID:
PHY Device Name/ID:
Buffer size: 32.768k
Decimation: 1
Parameters:

FMComms2/3/4 Source
IIO context URI: local:
LO Frequency: 2.4G
Sample rate: 2.084M
RF bandwidth: 20M
Buffer size: 32.768k
Quadrature: True
RF DC: True
BB DC: True
Gain Mode (RX1): Manual
Manual Gain (RX1)(dB): 64
Gain Mode (RX2): Manual
Manual Gain (RX2)(dB): 64
RF Port Select: A_BALANCED
Filter:
Filter auto: True

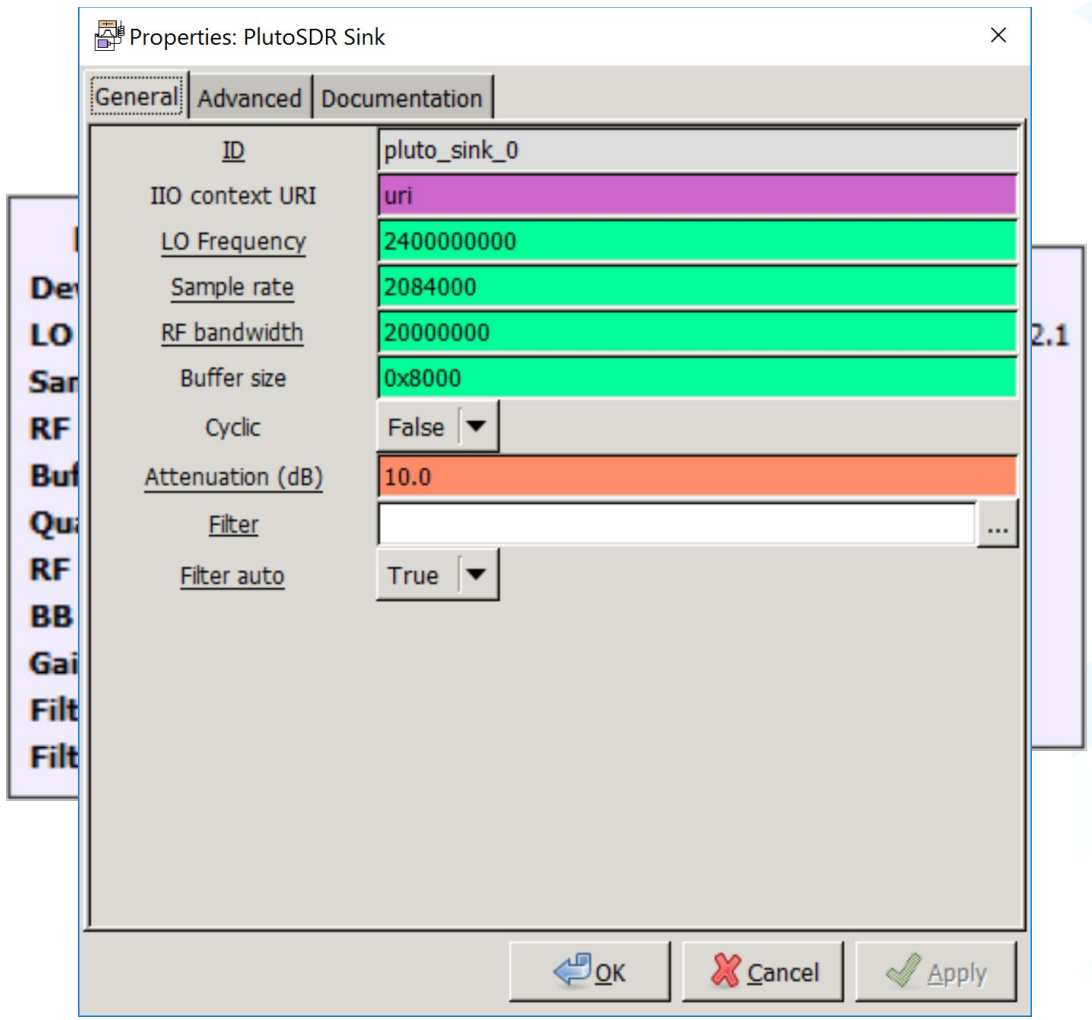
PlutoSDR Source
Device URI:
LO Frequency: 2.4G
Sample rate: 2.084M
RF bandwidth: 20M
Buffer size: 32.768k
Quadrature: True
RF DC: True
BB DC: True
Gain Mode: Manual
Manual Gain (dB): 64
Filter:
Filter auto: True

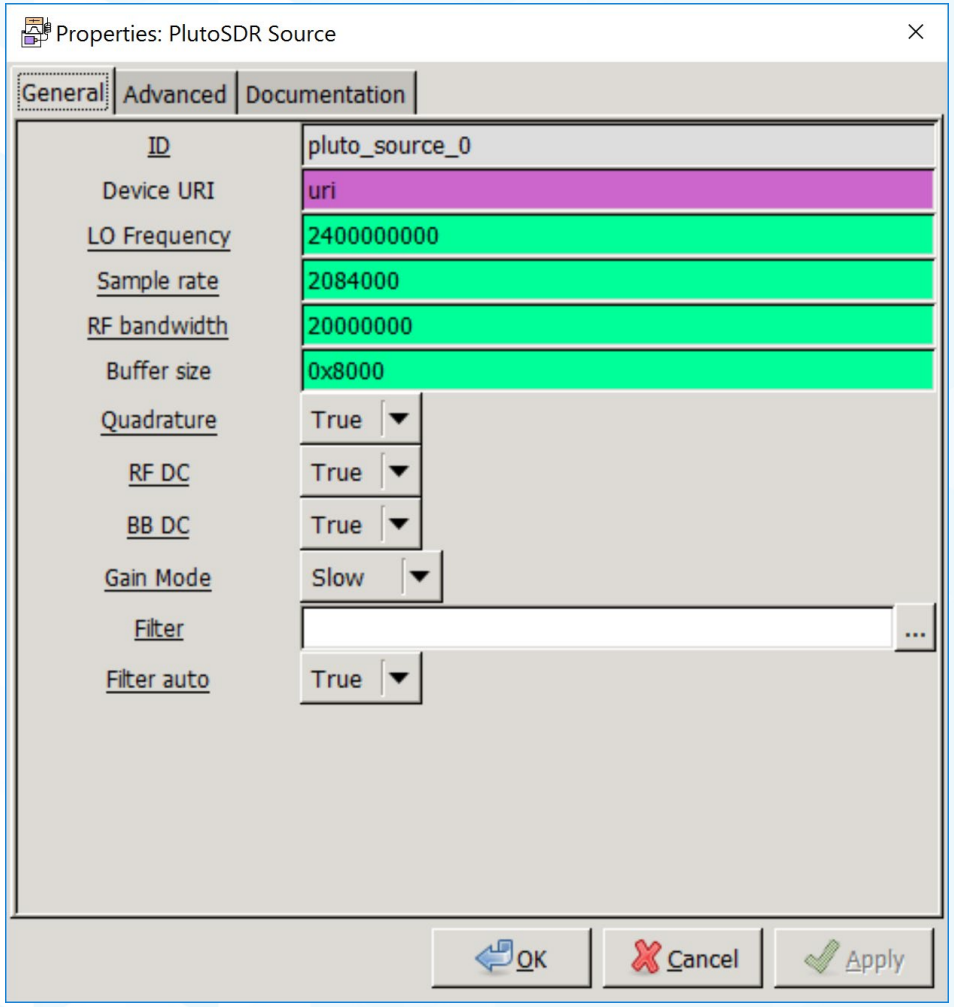
CUSTOM STREAM DEVICE



GRCon

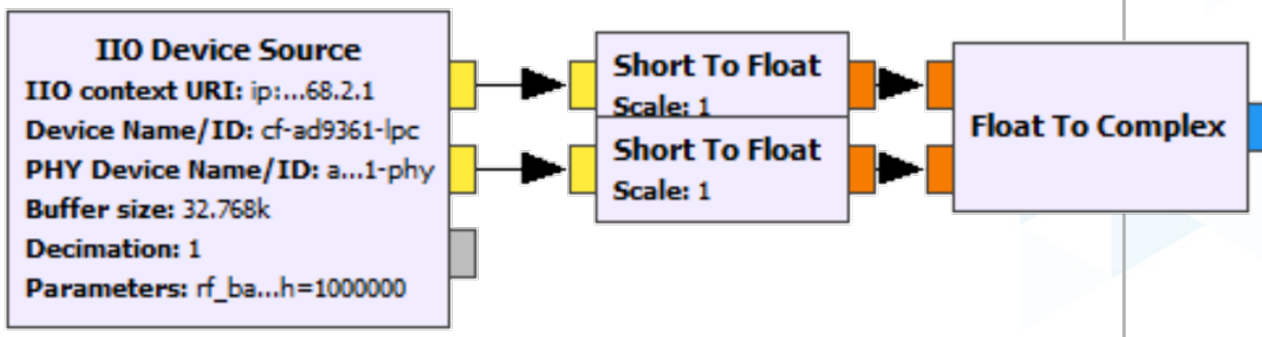
Using Pluto Blocks

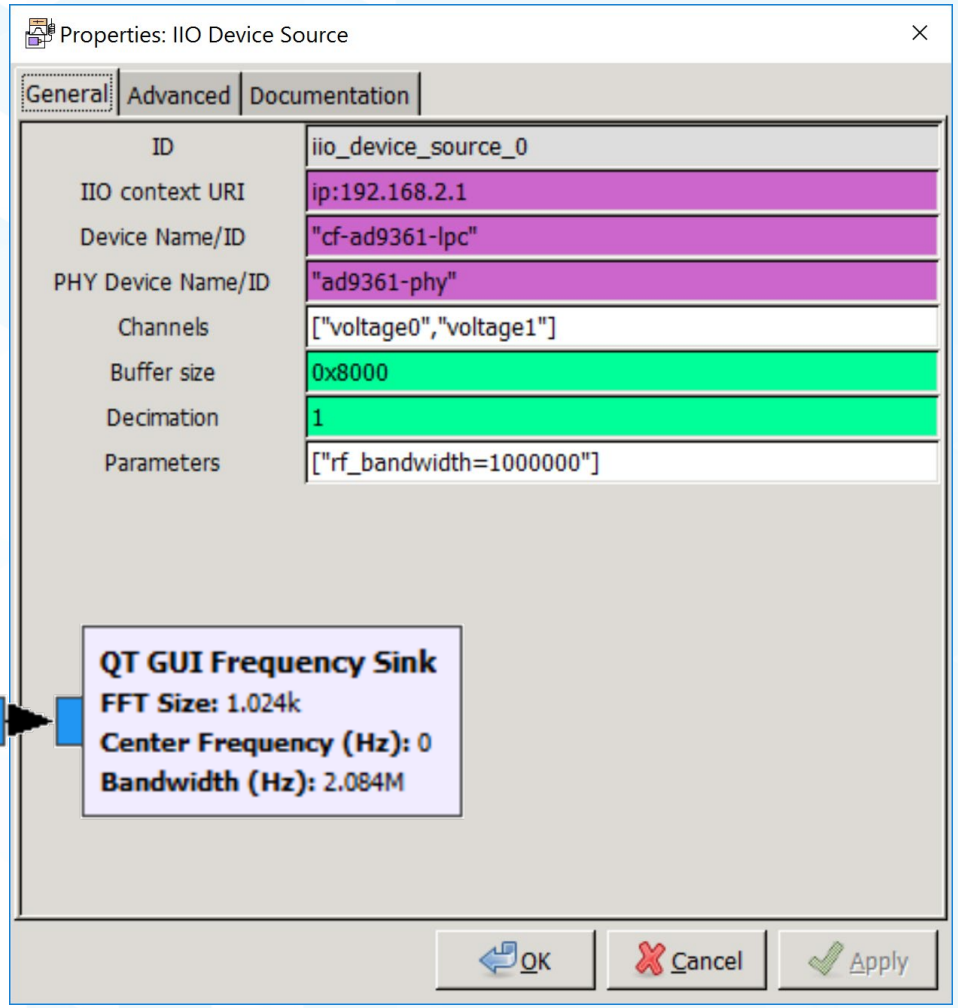




Stream (Device) Blocks

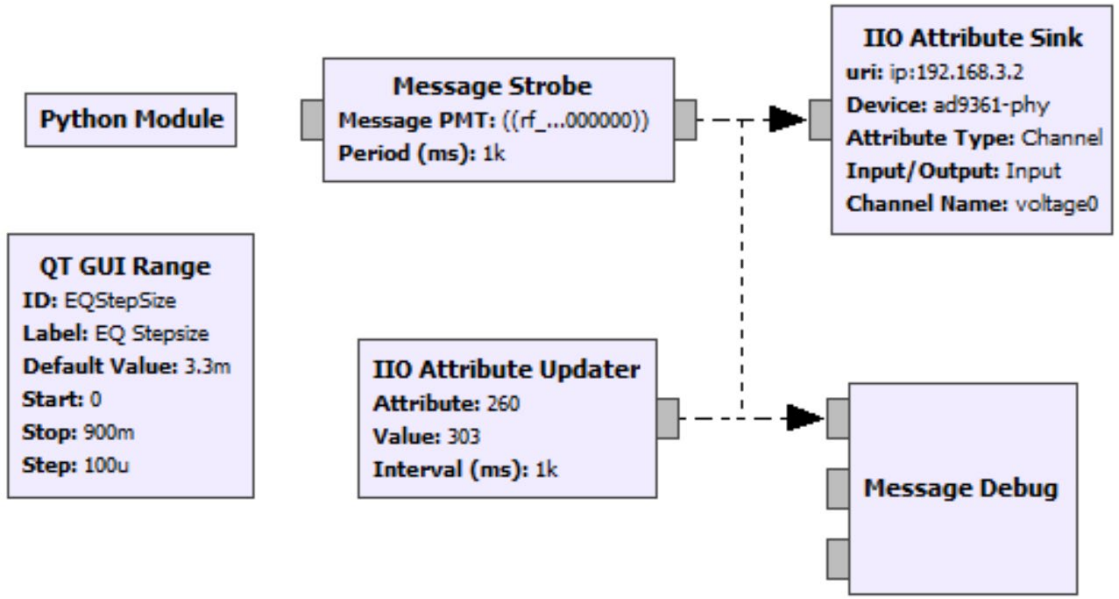
```
C:\Users\tcollins>iio_attr -q -u ip:192.168.2.1 -d
IIO context has 5 devices:
        iio:device3: cf-ad9361-dds-core-lpc, found 0 device attributes
        iio:device1: ad9361-phy, found 18 device attributes
        iio:device4: cf-ad9361-lpc, found 0 device attributes
        iio:device2: xadc, found 1 device attributes
        iio:device0: adm1177, found 0 device attributes
```

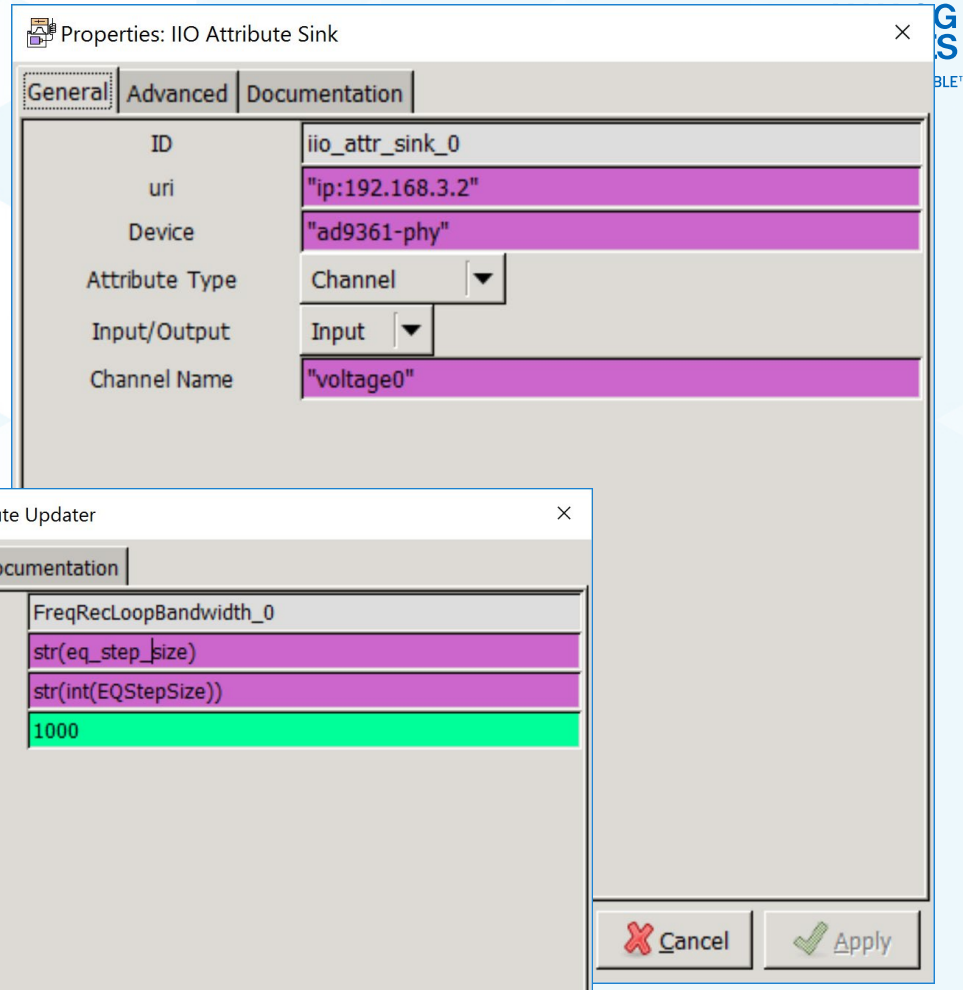


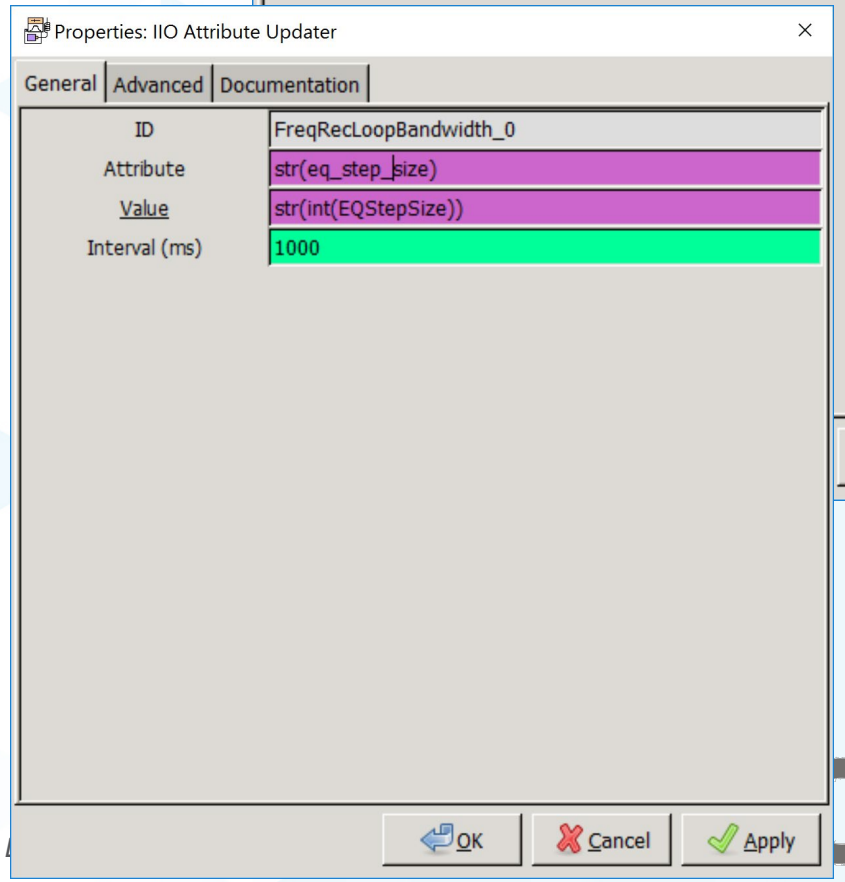


Using Attribute Sink

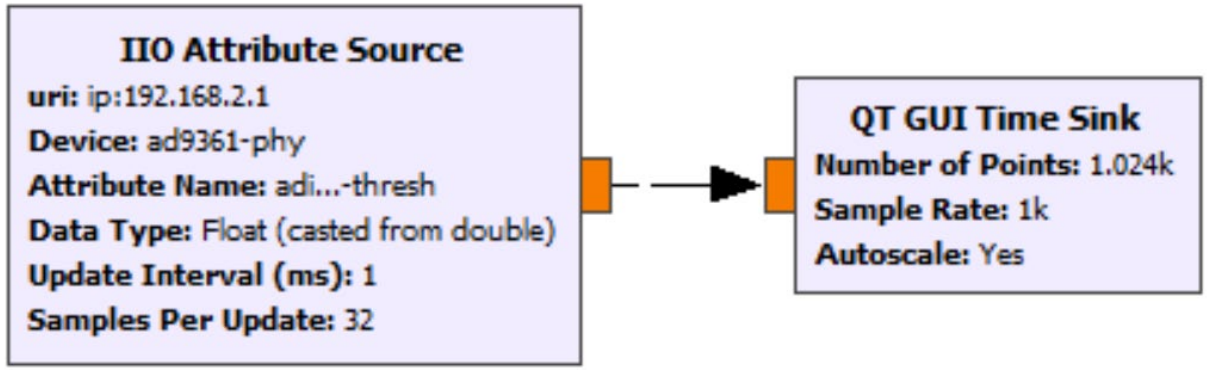
```
import pmtkey0 = pmt.intern("rf_bandwidth")
val0 = pmt.intern("23000000")
msg_dic = pmt.make_dict()
msg_dic = pmt.dict_add(msg_dic, key0, val0)
```

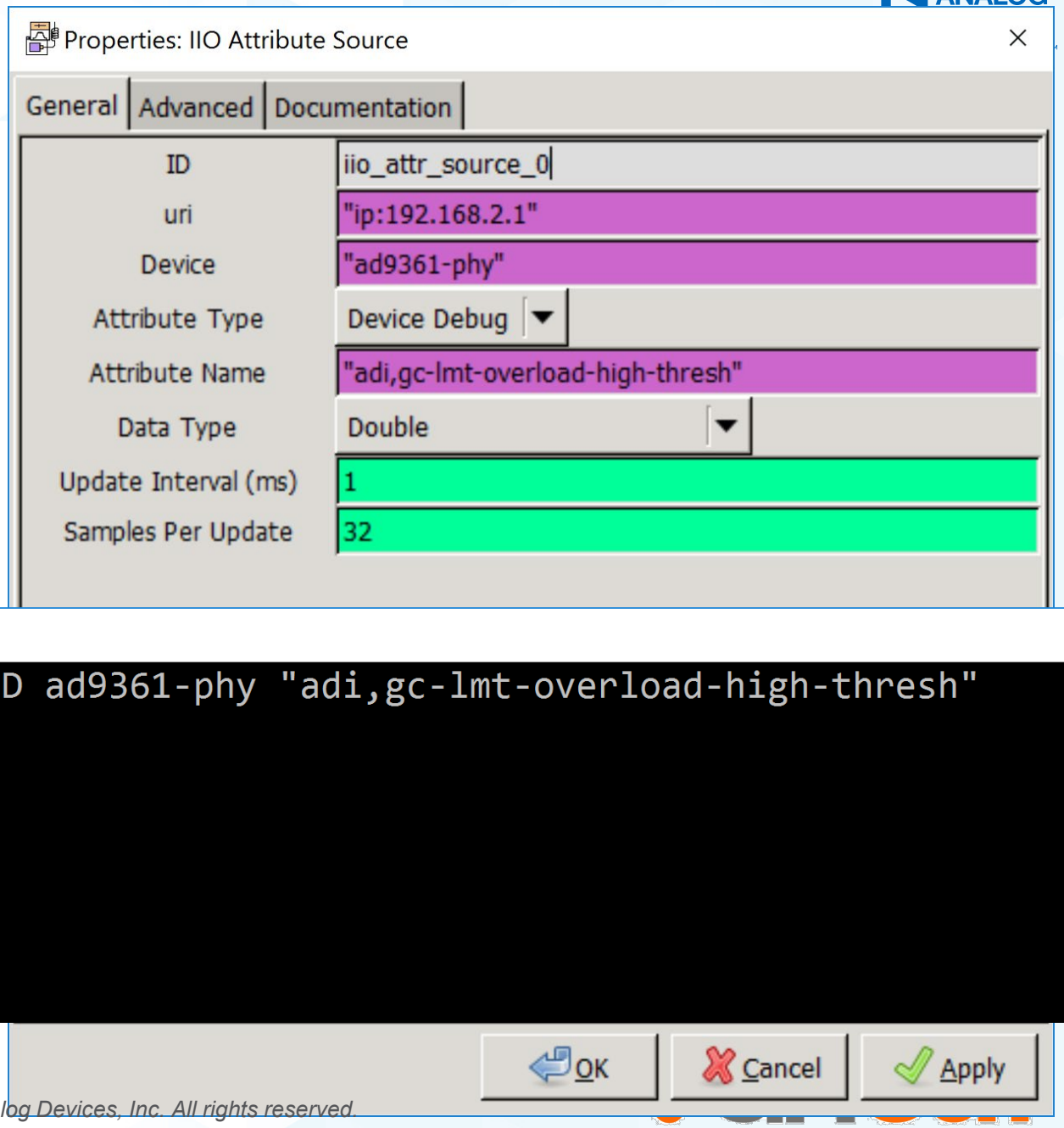


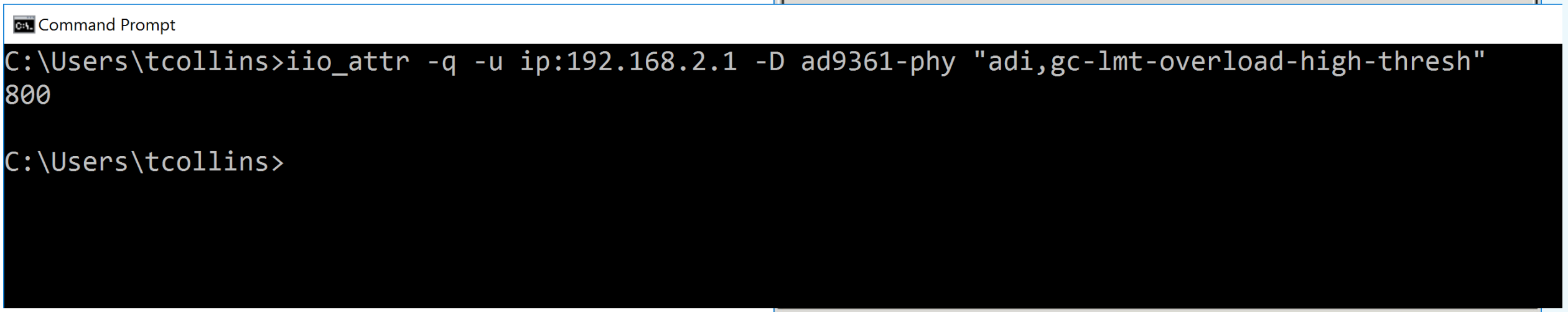




Using Attribute Source







Hands On Lab

USE LOOPBACK CABLE

- gr-iio in GNU Radio

- Lab file: **Intro-To-GNURadio-IIO-and-PlutoSDR.pdf**

http://www.iconarchive.com/show/noto-emoji-objects-icons-by-google/62807-radio-icon.html
http://www.streamlineicons.com
http://pixelkit.com

Thank you!

- For future support:
  - http://ez.analog.com
  - http://wiki.analog.com