Software Defined Radio
using the
Linux Industrial IO framework
- A Hardware Abstraction Layer -

Lars-Peter Clausen, Analog Devices

What is IIO?

- Industrial Input/Output framework
  - Not really just for Industrial IO
  - All non-HID IO
  - ADC, DAC, light, accelerometer, gyro, magnetometer, humidity, temperature, rotation, angular momentum, ...
- In the upstream Linux kernel since v2.6.32 (2009)
- Moved out of staging/ in v3.5 (2012)
- ~200 IIO device drivers (v3.19)
  - Many drivers support multiple devices

Why use IIO for SDR?

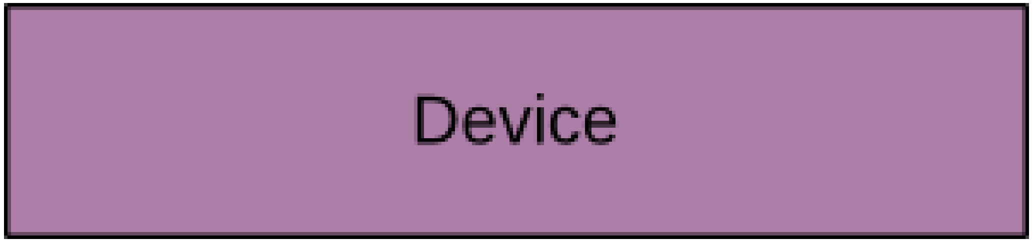
- Provides hardware abstraction layer
  - Allows sharing of infrastructure
  - Allows developers to focus on the solution
  - Allows application re-use

Why use IIO for SDR?

- Kernel drivers have low-level access to hardware
  - MMIO
  - Interrupts
- IIO provides fast and efficient data transport
  - From device to application
  - From application to device
  - From device to network/storage (soon)

IIO Framework

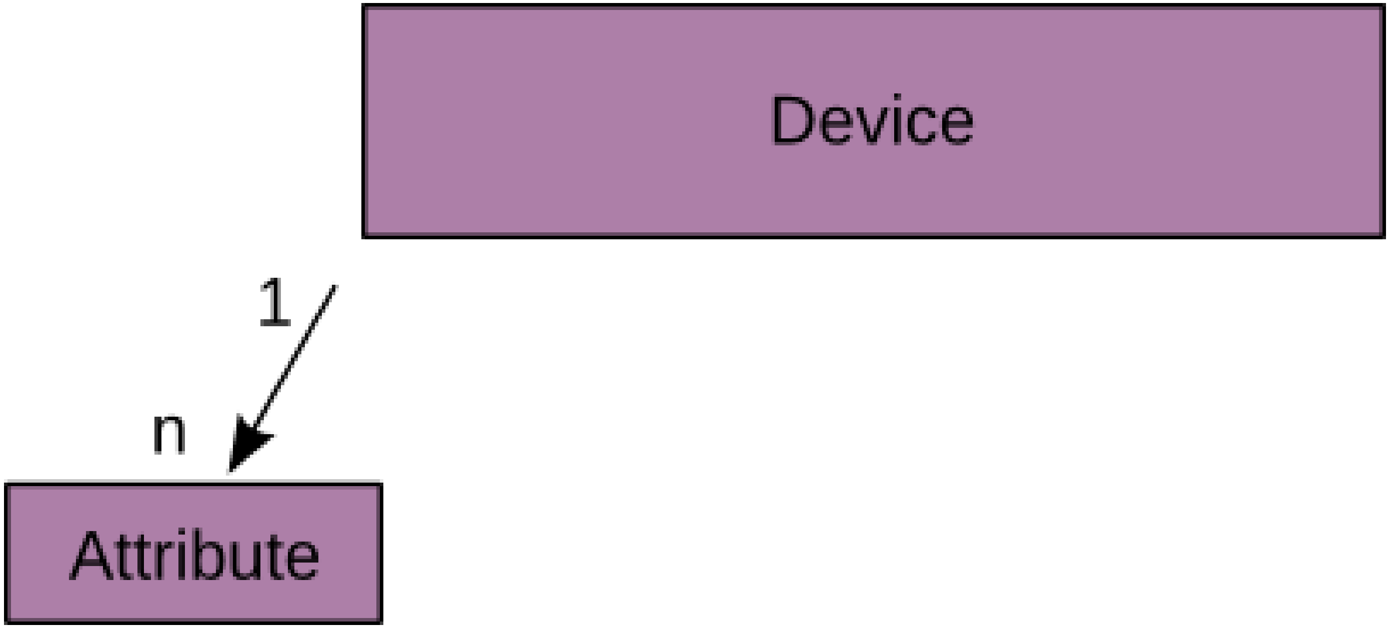
IIO – Devices



IIO – Devices

- Main structure
- Typically corresponds to a physical hardware unit
- Represented as directories in sysfs

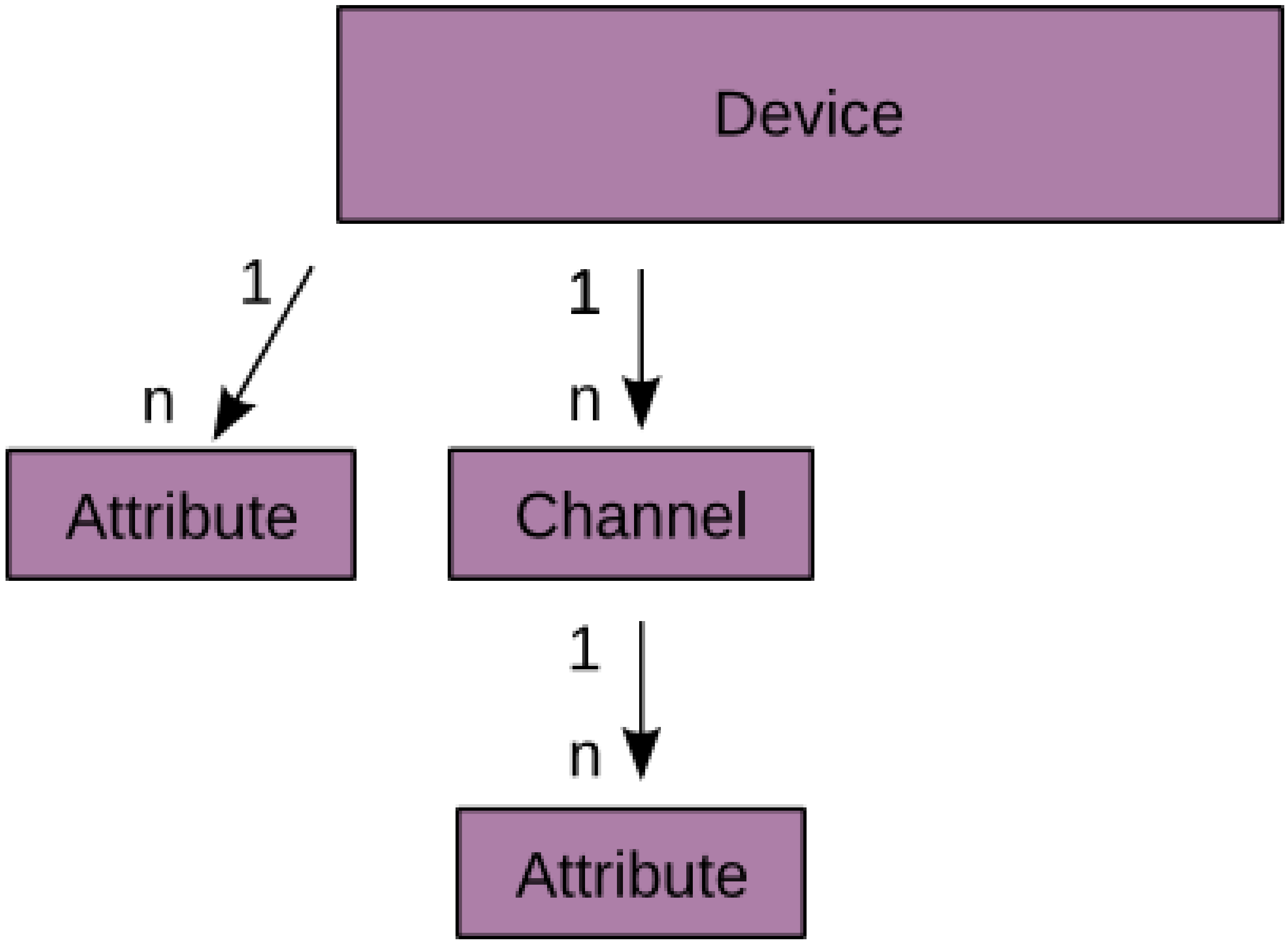
IIO – Attributes



IIO – Attributes

- Describe hardware capabilities
- Allow to configure hardware configuration
- Represented as files in sysfs

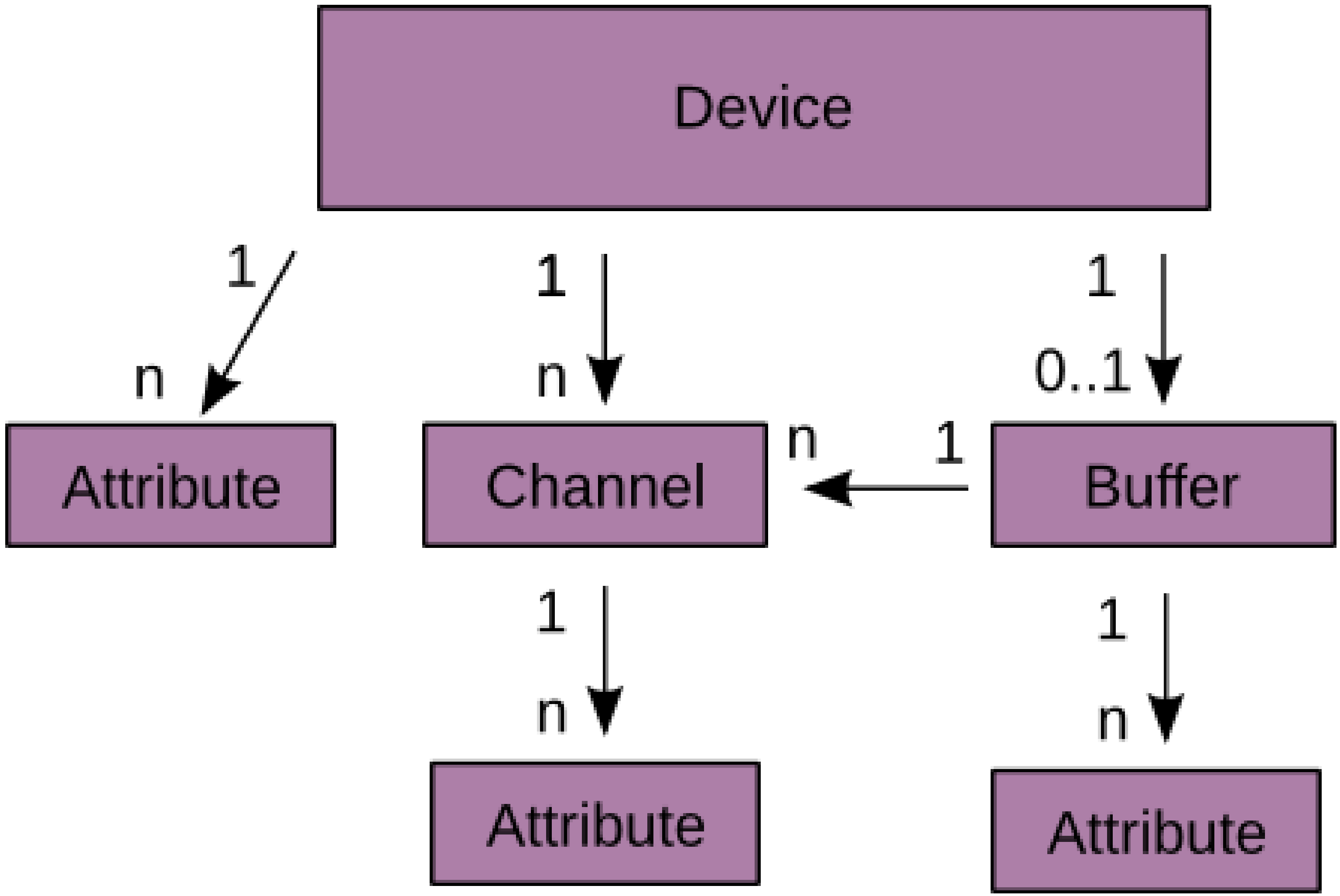
IIO – Channels



IIO – Channels

- Representation of a data channel
- Has direction, type, index and modifier
- Attributes provide additional information
  - scale, offset
  - Calibration data
  - Filters settings, hysteresis
  - ...

IIO – Buffers



Device
1
n
1
n
1
0..1
Attribute
Channel
n
1
Buffer
1
n
1
n
Attribute
Attribute

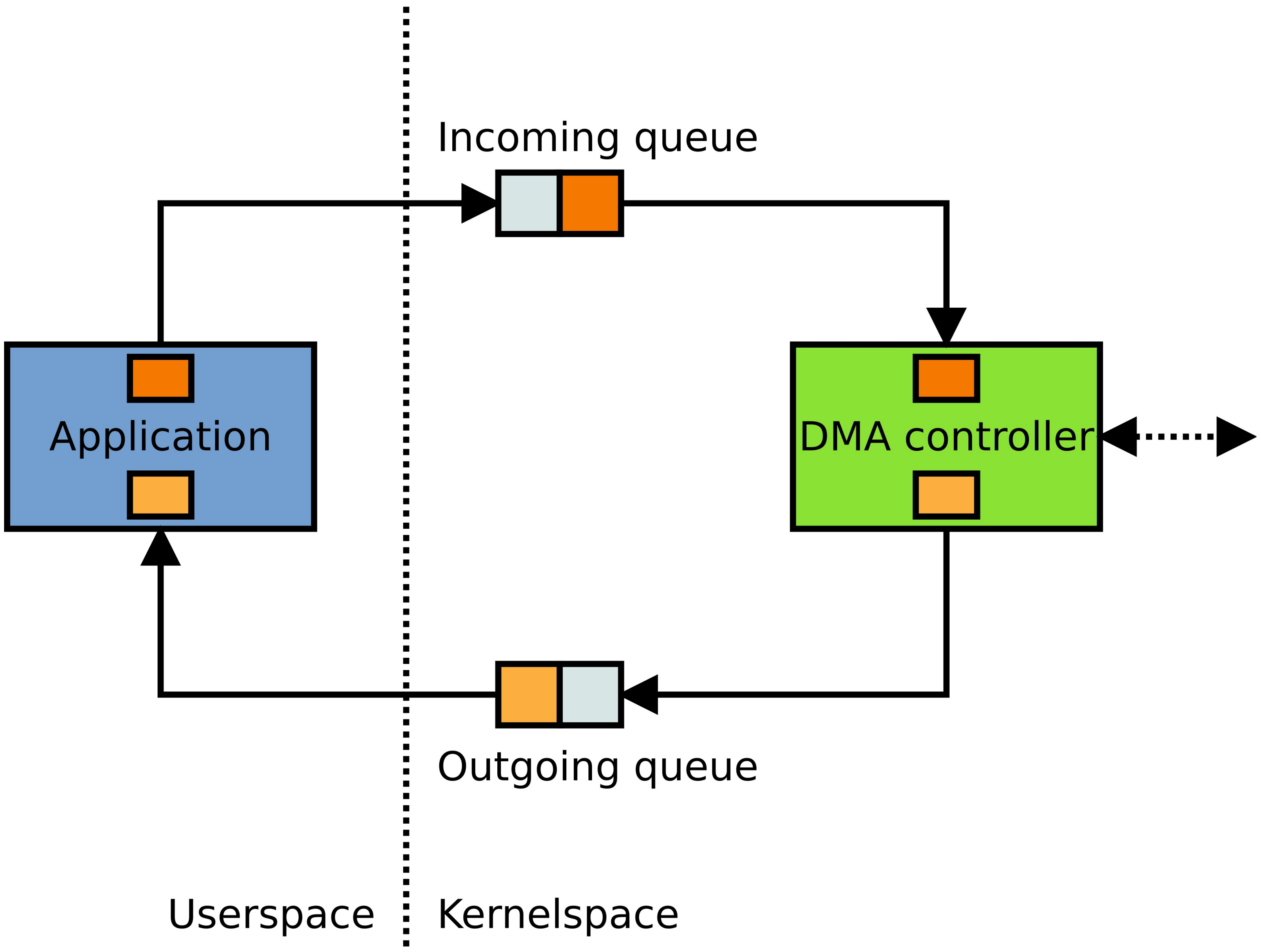
IIO – Buffers

- Used for continuous data capture/transmit
- Channels can be enabled/disabled
- Channels specify their data layout
- /dev/iio:deviceX allows read()/write() access
- Configuration using sysfs files
- Support for different buffer implementations
  - Software FIFO
  - DMA Buffer
  - Device specific buffer

IIO – DMA buffer

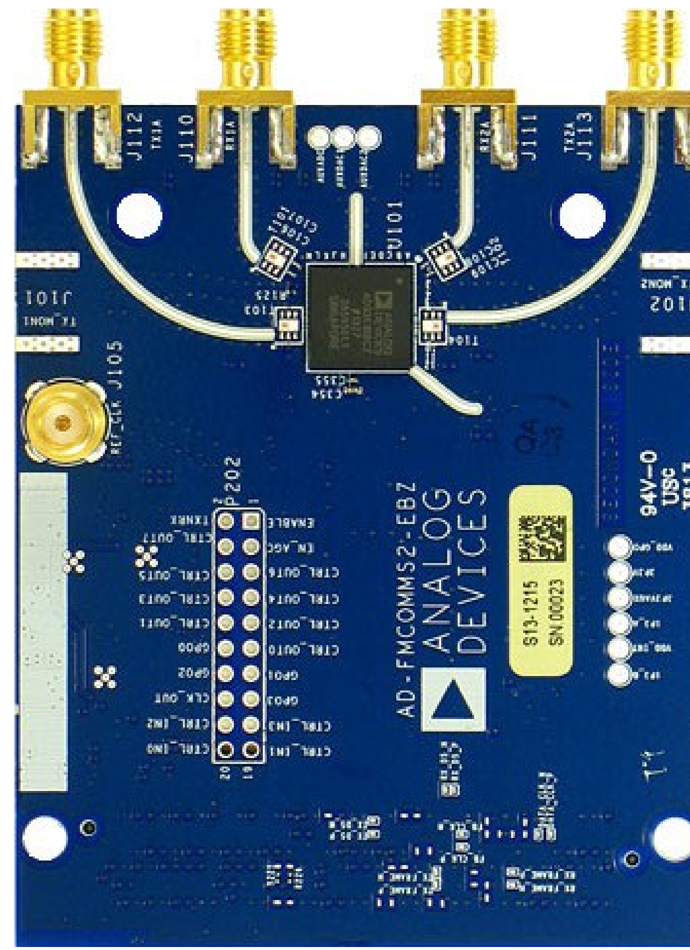
- DMA is used to copy data from device to memory
- mmap() is used to make data available in application
- Allows low overhead high-speed data capture
- Data is grouped into chunks (called DMA blocks) to manage ownership
  - Either application or driver/hardware owns a block

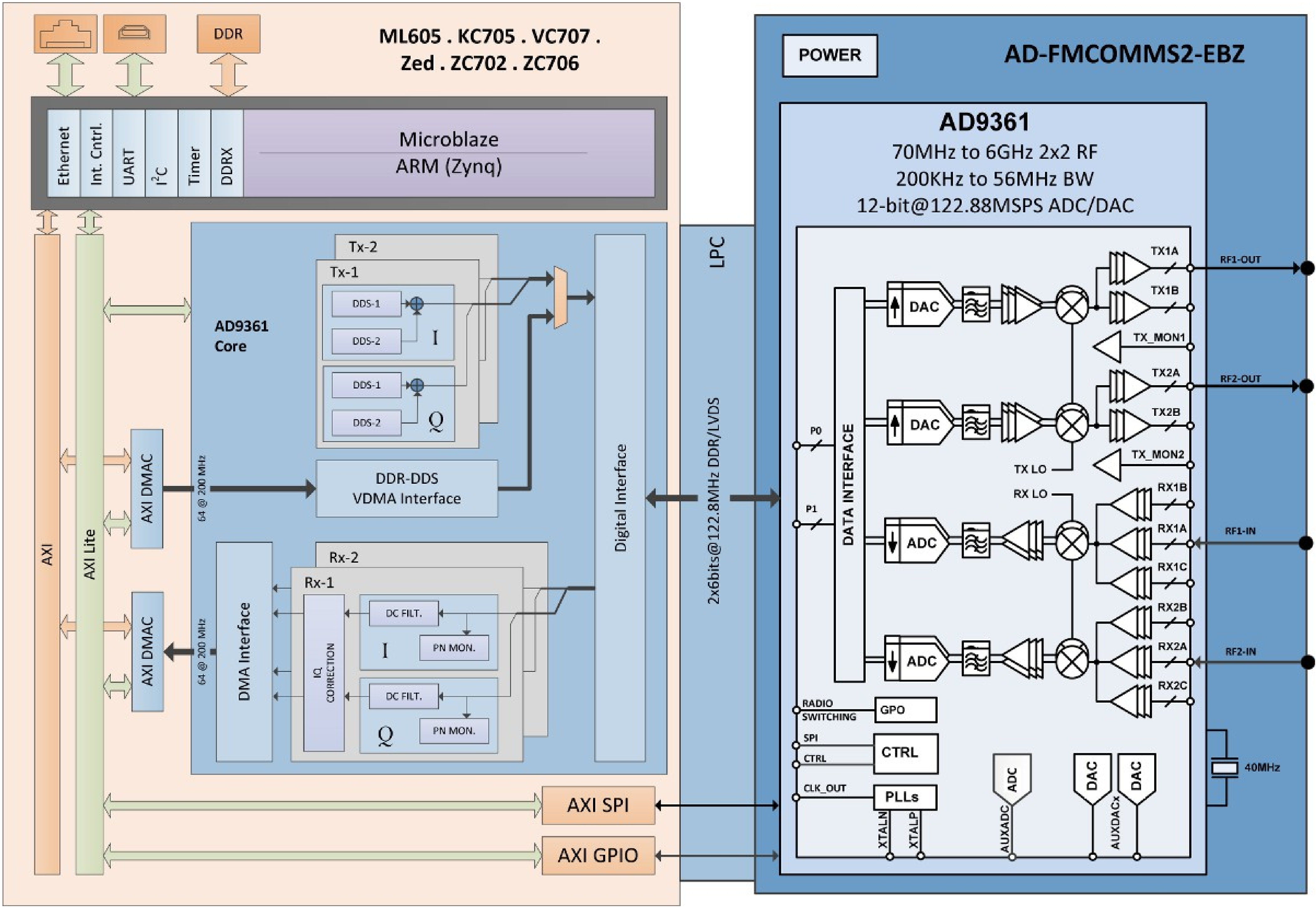
IIO – DMA buffer



Example – AD-FMCOMMS2-EBZ

- Software Defined Radio platform
- AD9361 Agile integrated transceiver
- 200 kHz - 56 MHz sample rate
- Tunable from 70MHz to 6GHz
- Full-duplex
- MIMO, 2x RX and TX
  - Each channel a set of 12-bit I and Q data




ML605 . KC705 . VC707 .
Zed . ZC702 . ZC706
DDR
Ethernet
Int. Cntrl.
UART
I2C
Timer
DDRX
Microblaze
ARM (Zynq)
AXI
AXI Lite
AD9361
Core
Tx-2
Tx-1
DDS-1
DDS-2
I
Q
AXI DMAC
64 @ 200 MHz
DDR-DDS
VDMA Interface
Digital Interface
Rx-2
Rx-1
DMA Interface
IQ CORRECTION
DC FILT.
PN MON.
AXI SPI
AXI GPIO
LPC
2x6bits@122.8MHz DDR/LVDS
POWER
AD-FMCOMMS2-EBZ
AD9361
70MHz to 6GHz 2x2 RF
200KHz to 56MHz BW
12-bit@122.88MSPS ADC/DAC
DATA INTERFACE
P0
P1
DAC
ADC
TX1A
TX1B
TX_MON1
TX2A
TX2B
TX_MON2
TX LO
RX LO
RX1B
RX1A
RX1C
RX2B
RX2A
RX2C
RF1-OUT
RF2-OUT
RF1-IN
RF2-IN
RADIO SWITCHING
GPO
SPI
CTRL
CLK_OUT
PLLs
XTALN
XTALP
AUXADC
AUXDACx
40MHz

Example – AD-FMCOMMS2-EBZ

```
root@analog:/sys/bus/iio/devices# ls
iio:device0  iio:device1  iio:device2
iio:device3  iio:device4

root@analog:/sys/bus/iio/devices# cat */name
ad7291
ad9361-phy
xadc
cf-ad9361-dds-core-lpc
cf-ad9361-lpc
```

Example – AD-FMCOMMS2-EBZ

```
# ls iio\:device1/
in_voltage_filter_fir_en
in_voltage_gain_control_mode_available
in_voltage_rf_bandwidth
in_voltage_rf_dc_offset_tracking_en
in_voltage0_gain_control_mode
in_voltage0_hardwaregain
in_voltage0_rssi
...
out_voltage_filter_fir_en
out_voltage0_hardwaregain
out_voltage0_rssi
...
filter_fir_config
...
in_temp0_input2
```

Example – AD-FMCOMMS2-EBZ

```
# ls iio\:device4/
buffer
in_voltage0_calibbias
in_voltage0_calibscale
in_voltage1_calibphase
in_voltage_sampling_frequency
in_voltage0_calibphase
in_voltage1_calibbias
in_voltage1_calibscale
name
scan_elements
```

Example – AD-FMCOMMS2-EBZ

```
# ls iio\:device4/buffer/
enable
length

# ls iio\:device4/scan_elements/
in_voltage0_en
in_voltage0_index
in_voltage0_type
in_voltage1_en
in_voltage1_index
in_voltage1_type
```

Plumbing Layer

libiio

- High level C interface to IIO
- Abstracts away low level details of IIO kernel ABI
- Transparently handles Low-Speed and High-Speed devices
  - Uses high speed interface when available

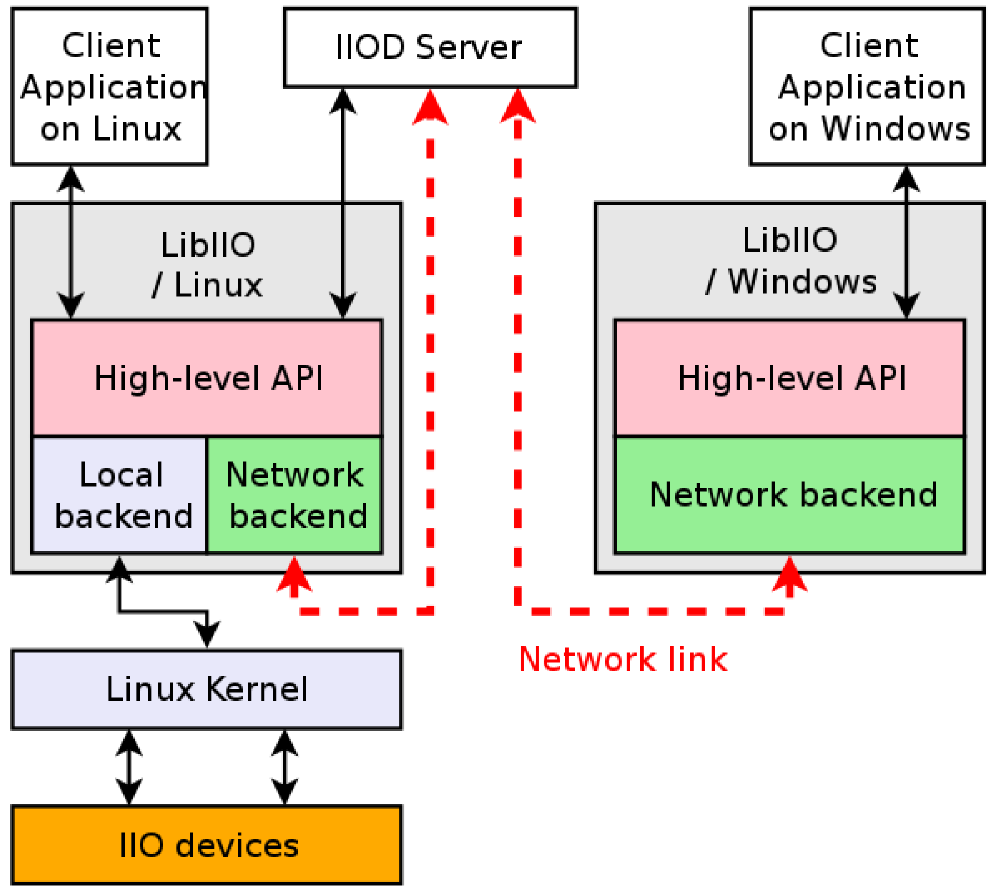
libiio

- Multiple backends
  - Local, directly using the IIO ABI
  - Network, uses network protocol to talk (remote) server (iiod)
  - Debug, fake devices for testing
- Bindings for python, C#, (C++)
- Cross platform (Linux, Windows)

iiod

- Multiplexing between multiple readers/writers
- Support for remote clients (via TCP/IP)
- Applications do not need system level privileges
- Transparent from the applications point of view

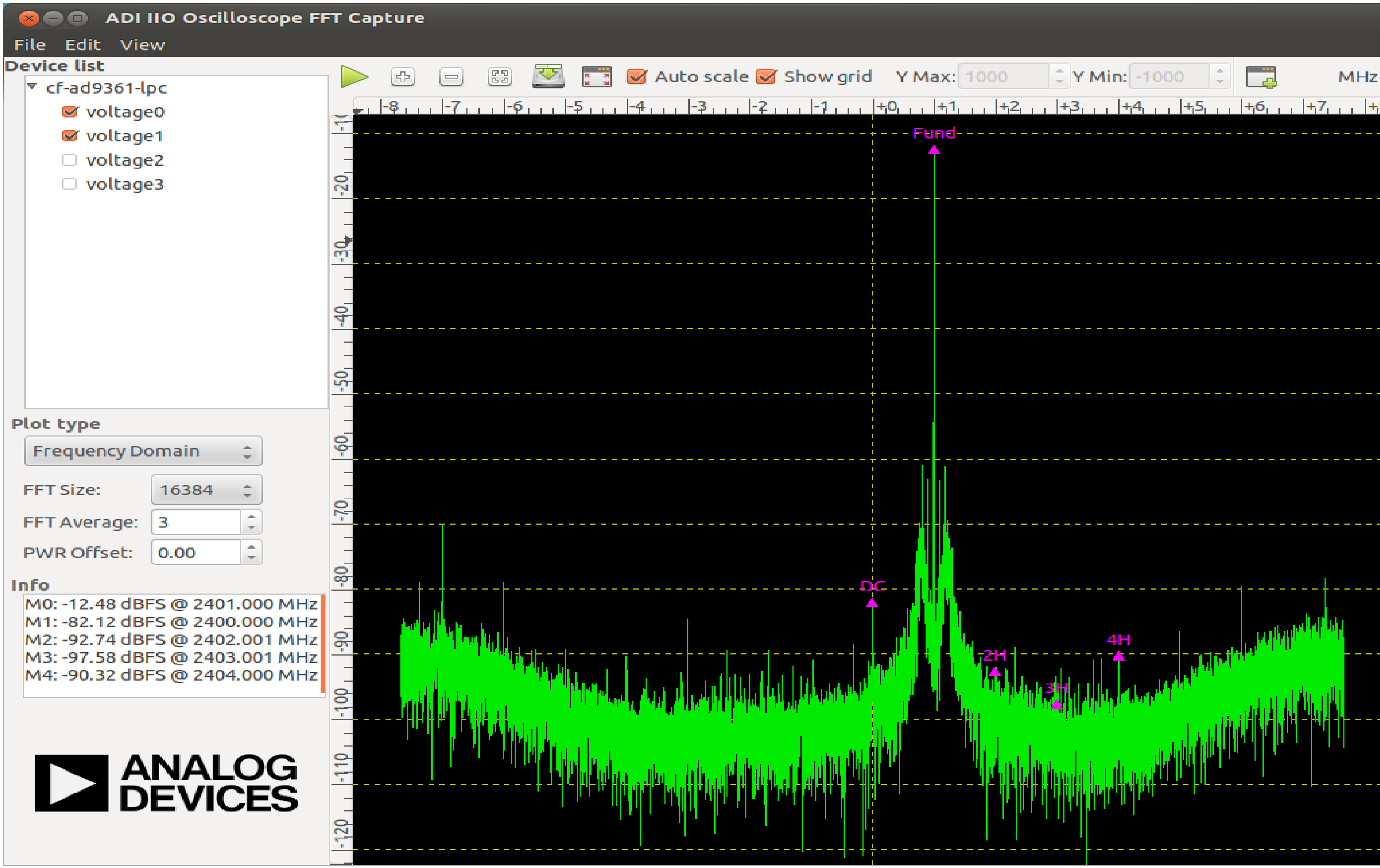
iiod & libiio


Client Application on Linux
IIOD Server
Client Application on Windows
LibIIO / Linux
High-level API
Local backend
Network backend
LibIIO / Windows
High-level API
Network backend
Linux Kernel
IIO devices
Network link

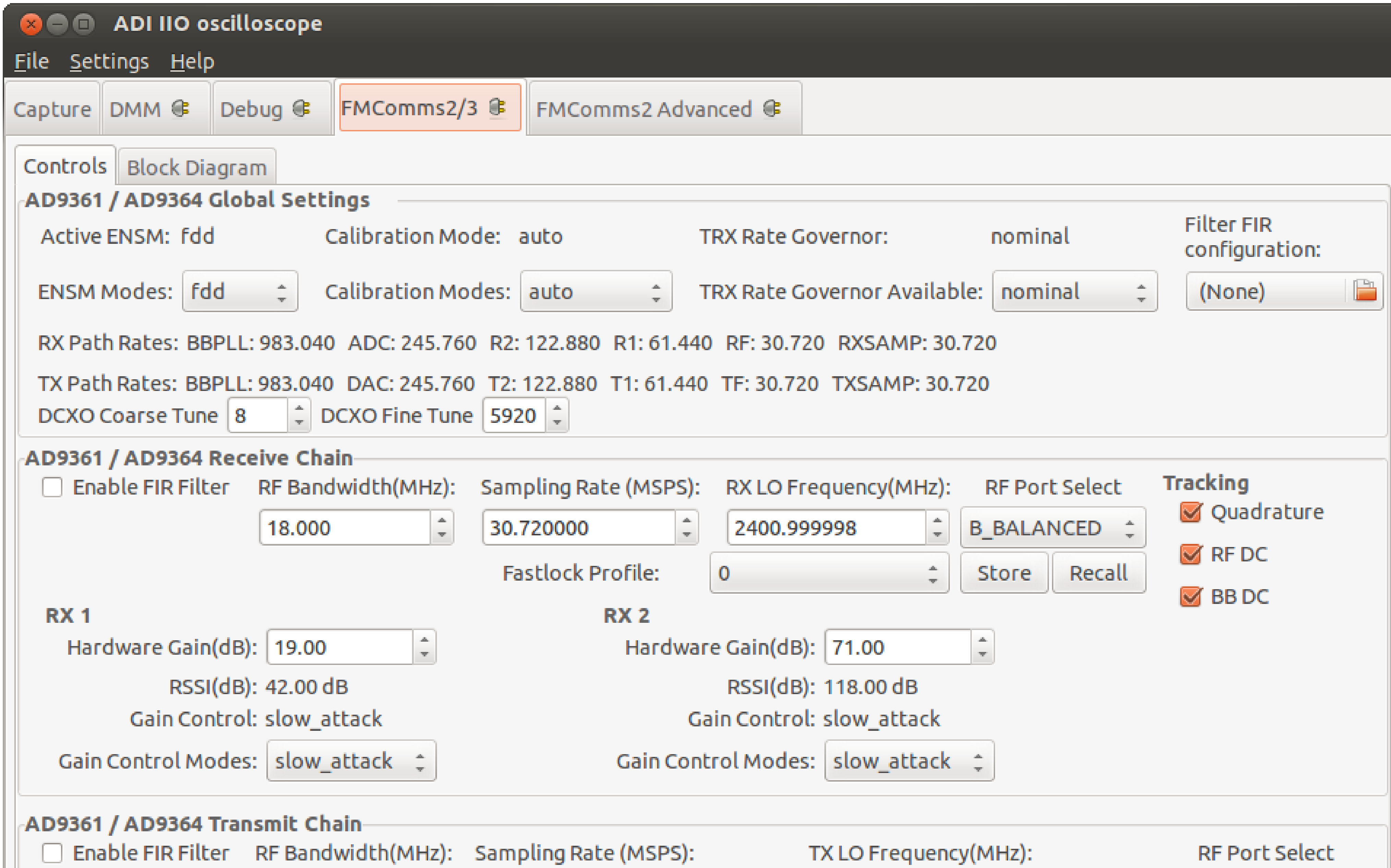
IIO Scope

- Capture and display data
  - Time domain, frequency domain, constellation, cross-correlation
  - Markers
- Plug-in system for easy configuration GUIs
- Custom math operations (experimental)

IIO Scope – Capture Window



IIO Scope – Plugins



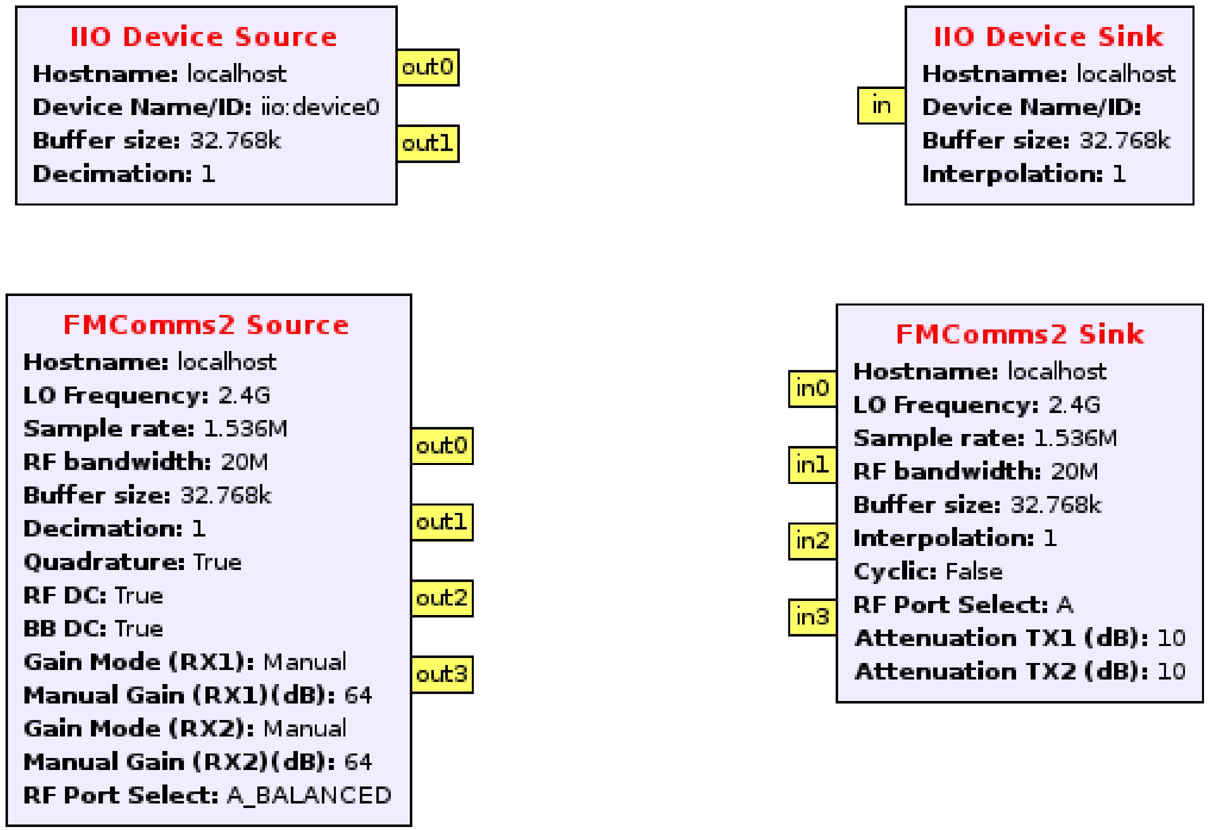
GNU Radio Plugin

- Two base classes
  - IIO Sink, Transmit data to a IIO device
  - IIO Source, Receive data from a IIO device
- Can select device and inputs/outputs
- Built-in support for Interpolation/Decimation

GNU Radio Plugin

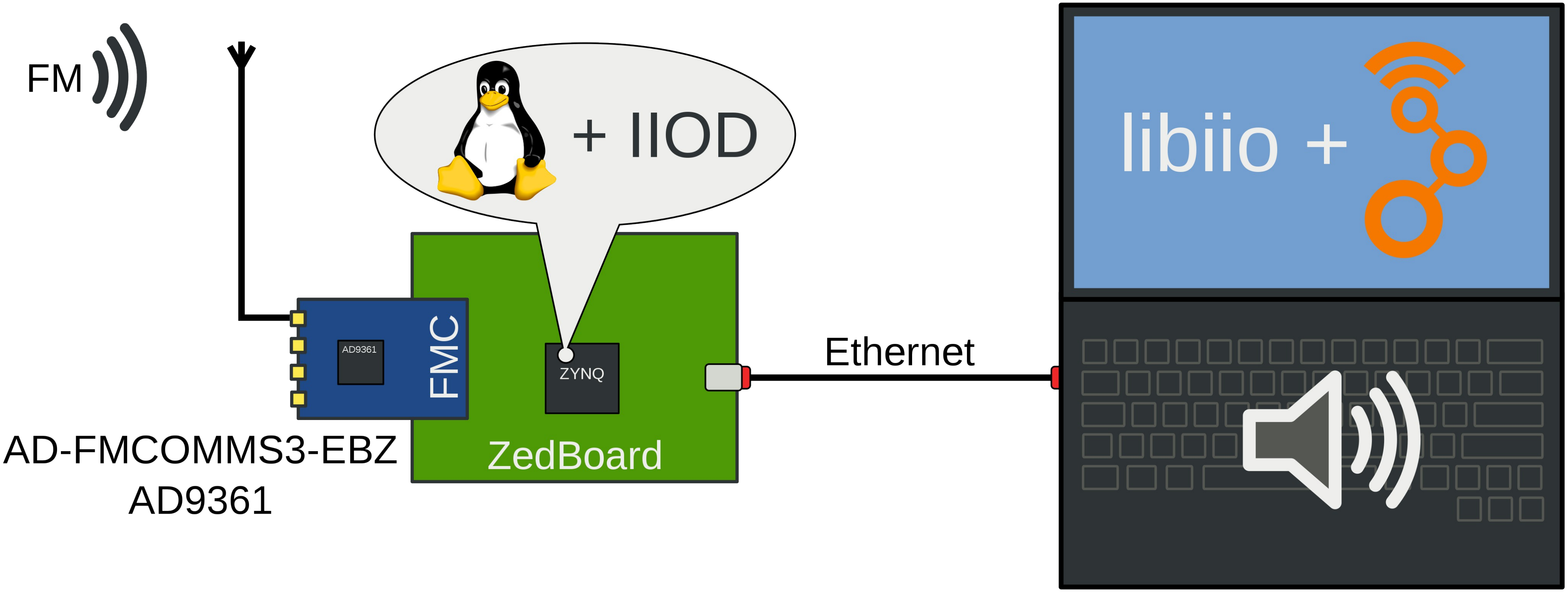
- Possible to subclass IIOSink/IIOSource
  - e.g. to implement device specific specialization
    - GUI
    - Setting attributes
  - Current examples:
    - FMCOMMS2 Sink
    - FMCOMMS2 Source

Gnu Radio Plugin



Demo

- FM radio receiver



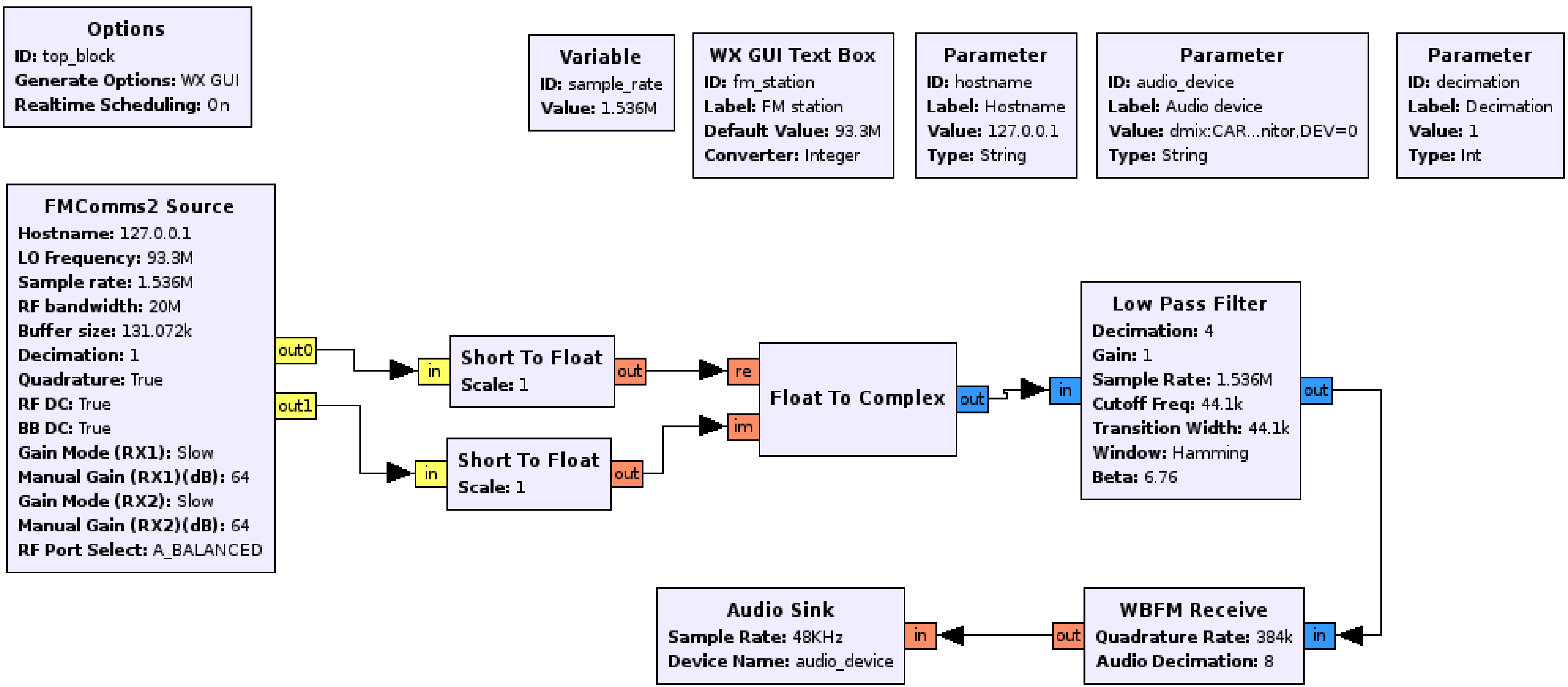
Demo Setup

- AD-FMCOMMS3-EBZ (AD9361)
- Zed Board (ZYNQ FPGA) running Linux with AD9361 IIO driver and IIOD
- Laptop running GNU Radio with the IIO plugin
- Laptop connected to ZED board via Ethernet

Demo Flow

- Capture data in the FM radio spectrum
- Stream it from the ZED board to the Laptop
- Decode the FM radio stream in GNU radio
- Playback FM radio on the Laptops speaker

Demo GNU Radio Canvas



Live Demo

Further information

- https://github.com/orgs/analogdevicesinc
  - https://github.com/analogdevicesinc/libiio
  - https://github.com/analogdevicesinc/iio-oscilloscope
  - https://github.com/analogdevicesinc/linux
  - https://github.com/analogdevicesinc/gnuradio
- http://wiki.analog.com/resources/tools-software/linux-software/libiio_internals
- http://analogdevicesinc.github.io/libiio/
- http://wiki.analog.com/resources/tools-software/linux-software/iio_oscilloscope
- https://wiki.analog.com/resources/tools-software/linux-software/gnuradio
- https://archive.fosdem.org/2012/schedule/event/iio.html
- http://events.linuxfoundation.org/sites/events/files/slides/iio_high_speed.pdf

Q/A

IIO – Device Graph (Future)

- Use media controller framework to expose device topology
  - Allows userspace to auto-discover processing pipeline
  - Better support for standard components

IIO – Zero Copy (Future)

- Use vmsplice and friends to provide zero copy
  - High-speed network streaming without CPU interaction
  - High-speed disk writes/reads without CPU interaction