SDR Driver and API options for the LimeSDR ecosystem and beyond

Lime Microsystems| FPRF company

Guildford, Surrey, United Kingdom

Sept 2017

Introductions: Josh Blum

Projects and open-source work

- GRC – GNURadio companion
- UHD - drivers, firmware, FPGA design
- VOLK – code generation + arch selection
- Maintainer SoapySDR + Pothos
- LimeSDR crowd funding campaign
- MyriadRF packaging support
- http://www.joshknows.com/projects

Embedded Engineer Skylark Wireless

Last mile wireless broadband solutions: Developing 5G communications hardware for rural and other under-served communities based on multi-user MIMO technology.

- http://www.skylarkwireless.com/

SDR Drivers/APIs

The ***boring*** part of SDRs

- Tedious APIs and layers
- Language choices etc...
- Documentation: Whats that?
- Compilers + dependencies
- Debugging: thats fun

The ***good***: why we do it ✓

- A good driver encapsulates functionality in a way that saves developer time and confusion
  - Set my gain in **dB** and my frequency in **Hz** – not register 0x24 = 0x3 << 3
  - Give me samples and flags – not bit field packing and magic offsets
- The human brain: *memory allocation error*
  - Layers give us the ability to split problems into manageable pieces with defined boundaries
- Code duplication? Ctrl+C, Ctrl+V, modify, repeat
  - Abstraction lets us write applications once – all the while supporting many similar devices

Soapy SDR: Motivation

Lime microsystems

A problem to solve (2014)

- I need to make a generic SDR support block
  - And I want to support most/all SDR devices
- Many projects: A new SDR on the market
  - Ctrl+C, Ctrl+V, modify, repeat
- Gr-osmosdr is very good, very close
  - It is massive: libboost, gnuradio, volk
  - No streaming API (needs gr blocks)
  - Difficult stream time/burst controls
  - New SDR? Ctrl+C, Ctrl+V, modify

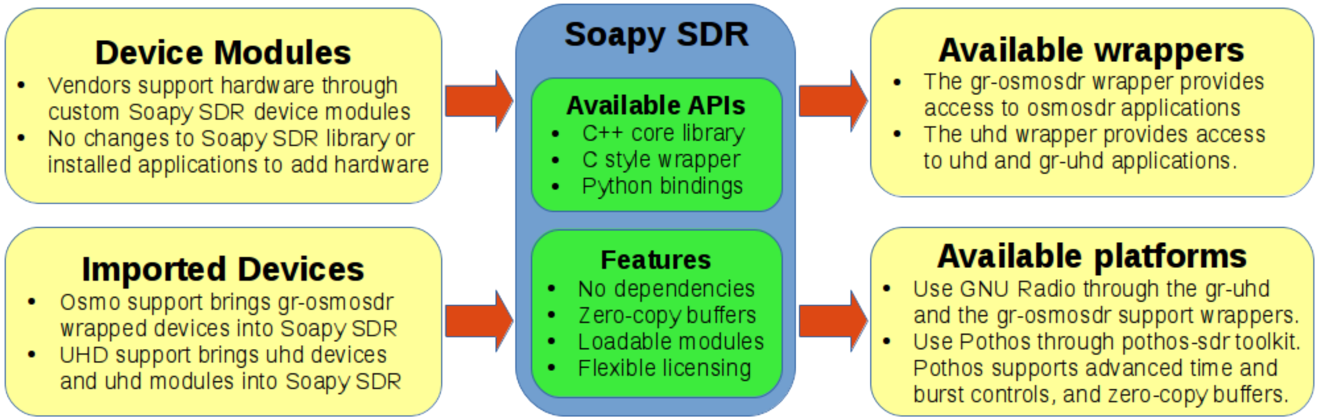
Soapy SDR: Design considerations

Designing a framework: requirements

- Make an API that anyone can use, not application specific
  - Generalized support for device enumeration, identification
  - Restful API for generalized SDR settings: frequency, gain, rates, filters, sensors…
  - Streaming API: read and write samples and metadata, stream status too
- Minimal dependencies for the core project
  - Just a compiler and make/cmake
- Modules/plugin architecture based (decoupling)
  - Load hardware support libraries at runtime
  - Do not recompile framework for new hardware
- Permissive licensing for commercial and open source

Soapy SDR: Basic Features

- C++/C and python API
- Very low boilerplate
  - CMake macro
  - Settings, Streaming, Registration.cpp - Overload the calls you need
- Modules for most devices: RTL, HackRF, USRP, AirSpy, LimeSDR…
- SoapyRemote – use any SDR over a network
- SoapyMultiSDR – N devices, 1 handle
- SoapyOsmo – wraps gr-osmosdr hardware support without gr dependencies
- https://github.com/pothosware/SoapySDR/wiki


Device Modules
Vendors support hardware through custom Soapy SDR device modules
No changes to Soapy SDR library or installed applications to add hardware
Imported Devices
Osmo support brings gr-osmosdr wrapped devices into Soapy SDR
UHD support brings uhd devices and uhd modules into Soapy SDR
Soapy SDR
Available APIs
C++ core library
C style wrapper
Python bindings
Features
No dependencies
Zero-copy buffers
Loadable modules
Flexible licensing
Available wrappers
The gr-osmosdr wrapper provides access to osmosdr applications
The uhd wrapper provides access to uhd and gr-uhd applications.
Available platforms
Use GNU Radio through the gr-uhd and the gr-osmosdr support wrappers.
Use Pothos through pothos-sdr toolkit. Pothos supports advanced time and burst controls, and zero-copy buffers.

Soapy SDR: Interesting uses

Unexpected uses/idioms

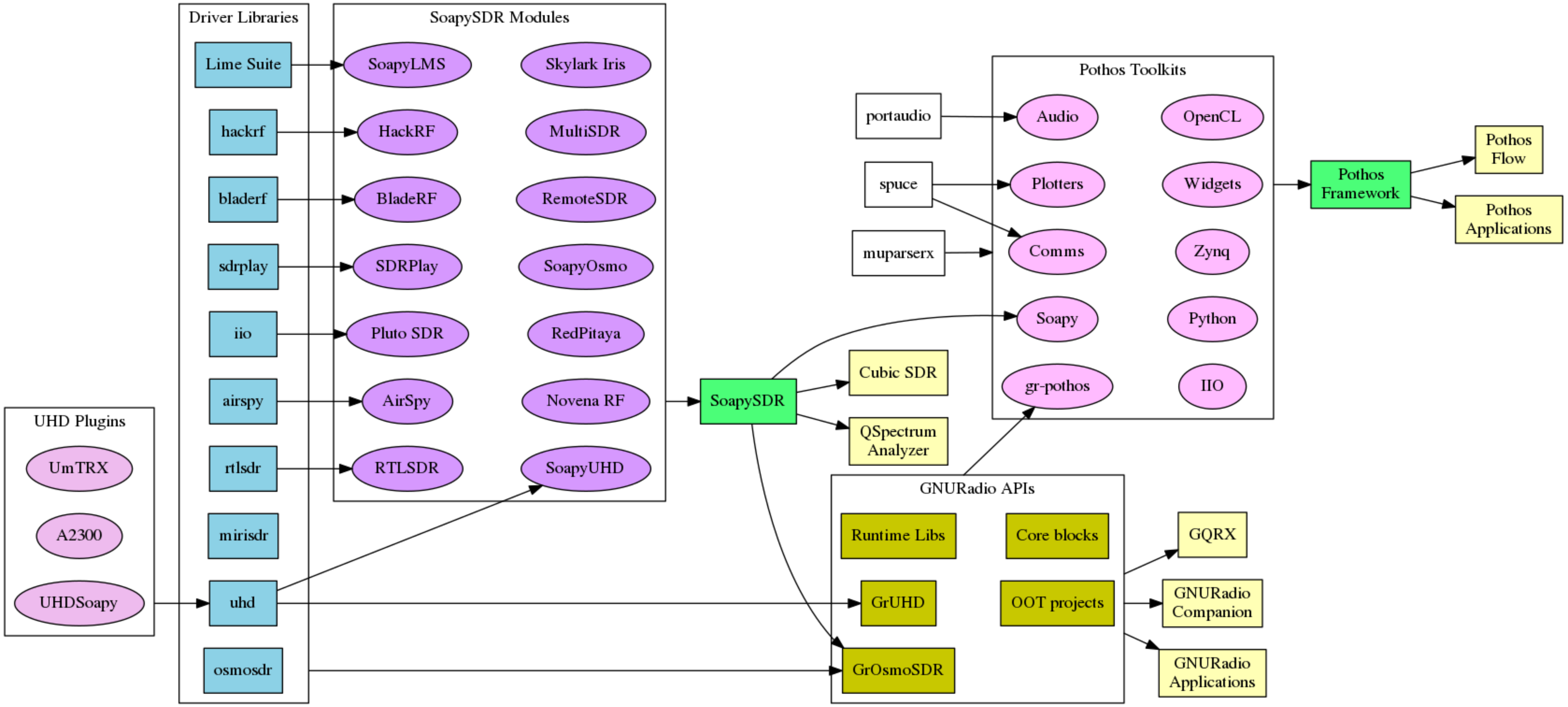
- Wrap entire HW support into SoapySDR module – No C API whatsoever
- Or bundle SoapySDR module with low-level driver: LimeSuite
- Not everything is sample streams: decoded packets, bounded arrays of bytes…
- Low level APIs: registers, SPI, I2C, UART, generic settings...
- SoapyRemote, but with custom streams: Zynq FPGA and Skylark Iris hardware

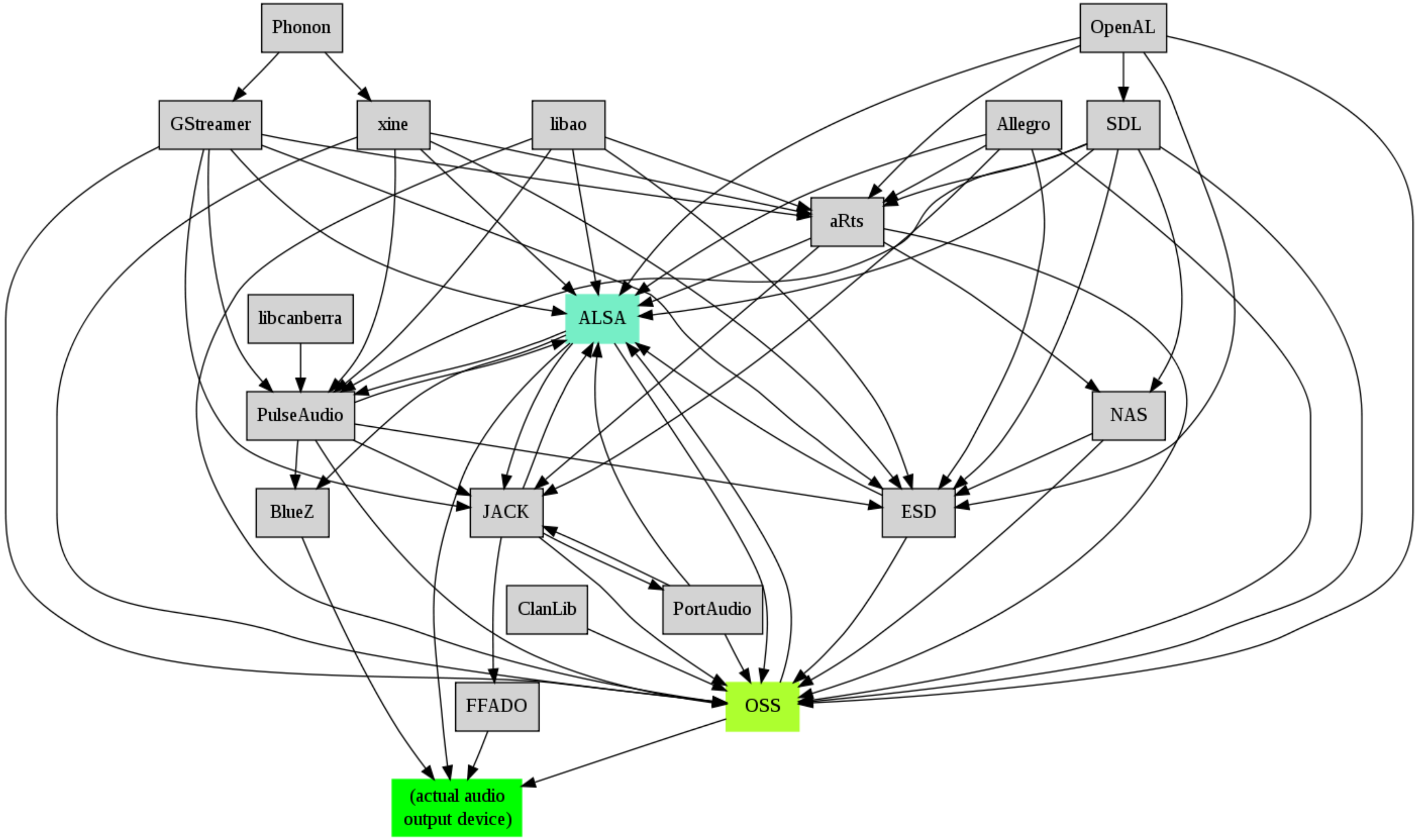
Closing the loop

- Gr-osmosdr has soapy support too
  - Anything SoapySDR works in GQRX, GRC, etc...
- UHDSoapy – support in UHD API
  - USRPs get remote device support
  - uhd_usrp_probe a RTLSDR :-)

Ecosystem of software

(*not complete (obviously (but kind of cool)))

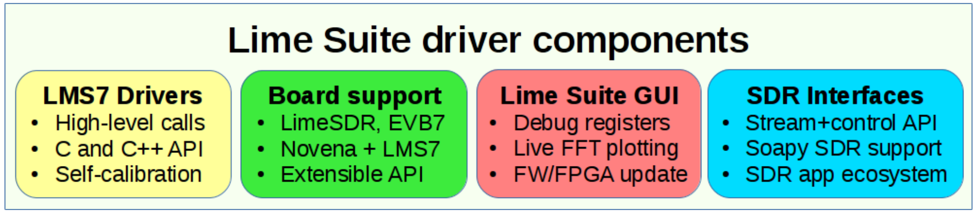




The Linux Audio Mess
Origin: Mike Melanson, http://blogs.adobe.com/penguin.swf/
Updated October 10, 2008

Lime Suite: Introduction

A driver for LimeSDR + much more



Lime Suite driver components

LMS7 Drivers
- High-level calls
- C and C++ API
- Self-calibration

Board support
- LimeSDR, EVB7
- Novena + LMS7
- Extensible API

Lime Suite GUI
- Debug registers
- Live FFT plotting
- FW/FPGA update

SDR Interfaces
- Stream+control API
- Soapy SDR support
- SDR app ecosystem

- LimeSDR + **other** devices featuring LMS7002M
- Reusable parts for developing with LMS7002M
  - LMS7002M driver: register abstraction and high level calls
  - Open FPGA designs projects and matching driver support
  - Mix and match custom hardware, fpga, and driver code
- Similar API for device enumeration + settings
  - High level API for generic devices based on LMS7002M
  - Python too: https://myriadrf.org/projects/pylms7002m/
  - Automatic support for devices under SoapySDR + friends
  - Device works in LimeSuite GUI for RFIC debugging

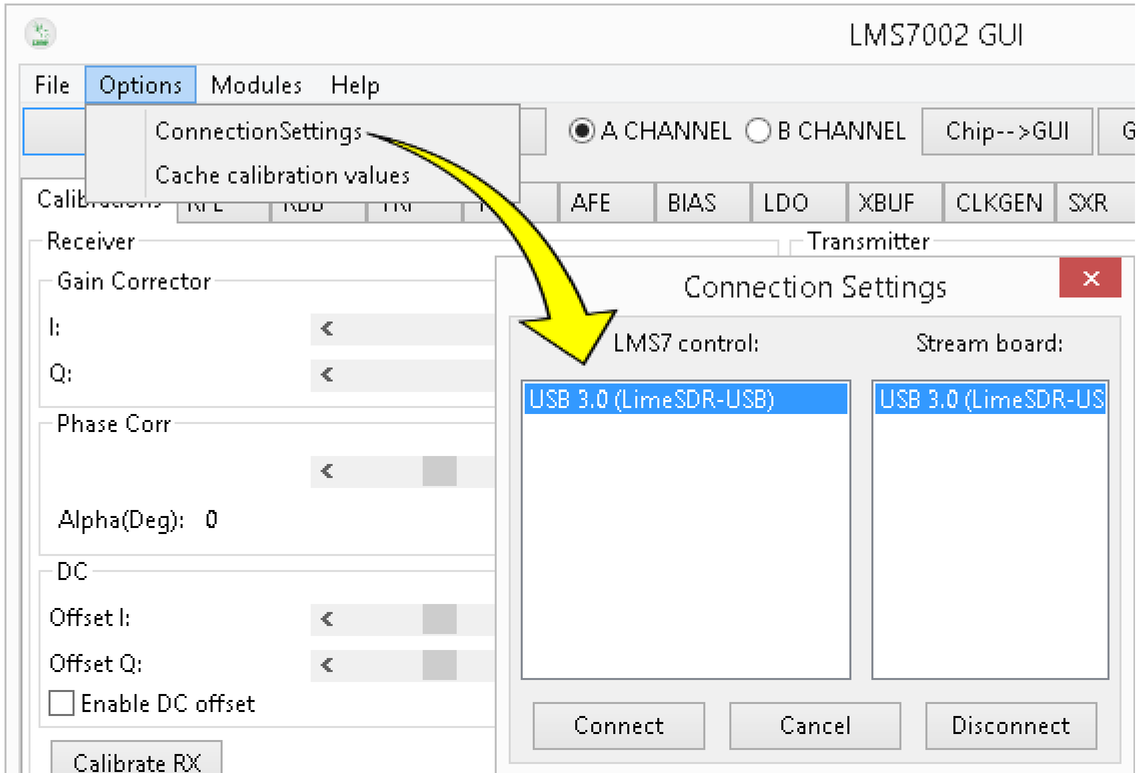
Lime Suite: Application components

LimeSuite C API

- #include <lime/LimeSuite.h>
- Full C API 100% in limesuite
- Enumerate, stream, configure
- Also hardware specific stuff
- Low level, FPGA programming

LimeSuite GUI

- Register dumps (debugging)
- Low level and high level controls
- Enumeration, firmware flashing
- FFT viewer and Tx waveforms



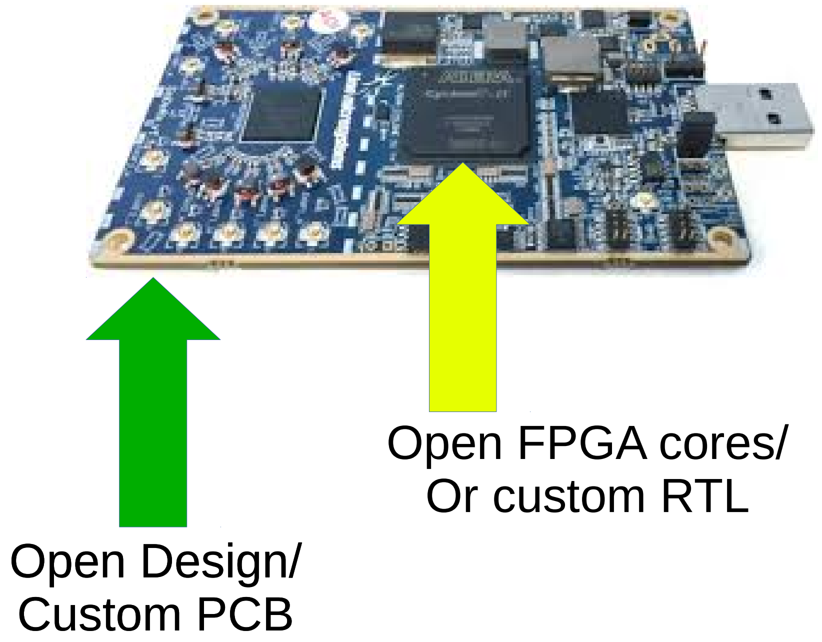
Lime Suite: Custom PCB + Drivers

Plugging into LimeSuite (c++)

- lime::IConnection + lime::ConnectionRegistry
- Device enumeration, register IO, streaming
- Tell LimeSuite how to talk to LMS7002M SPI
- Tell LimeSuite how stream Rx/Tx samples
- Yeah it works! C API, LimeSuite GUI, SoapyLMS7

And reusing LimeSDR FPGA cores

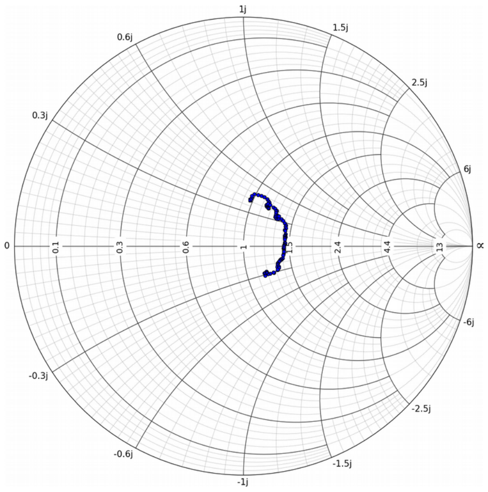
- Reuse existing FPGA cores (burst+time control)
- Inherit lime::LMS64CProtocol this time
- Tell LimeSuite how to talk to LMS7002M SPI
- R/W IO streams: High level timestamp samples
- Yeah it works! C API, LimeSuite GUI, SoapyLMS7



Lime Suite: Other items of interest

pyLMS700M

- Low level API – for python
  - https://myriadrf.org/projects/pylms7002m/
- VNA Example with pylms7002m
  - https://myriadrf.org/blog/lms7002m-python-package-vna-example/



LMS7002M embeddable C driver

- All C driver implementation, no dependencies
- Embed into another project: static lib, or directly
- Using it at SkylarkWireless for the Iris modules
- Drop it into a kernel module or micro-controller
- https://github.com/myriadrf/LMS7002M-driver

Software packaging @ MyriadRF

- Launchpad.net PPAs
  - https://launchpad.net/~myriadrf
- Ubuntu SNAP packages
  - https://github.com/myriadrf/snapcraft-sandbox
- Windows installer – PothosSDR
  - https://github.com/pothosware/PothosSDR/wiki

Get Involved: http://wiki.myriadrf.org/Packaging

Software packaging:

launchpad

- Launchpad.net builds and hosts deb packages for Ubuntu from source
- PPAs maintained at MyriadRF:
  - sudo add-apt-repository -y ppa:myriadrf/drivers
  - sudo add-apt-repository -y ppa:myriadrf/gnuradio
- Recent versions of Ubuntu releases and LTS releases
- Up to date hardware drivers, soapy modules, gnuradio, gr-osmosdr, others
- Sometimes backports, sometimes development branches
- Special thanks to Alexandru Csete: http://gqrx.de/
- Volunteers to test packages, make requests, and help maintain!

But sometimes debs can be difficult...

- Mixing with libs with /usr/local
- Dependencies on older ubuntu
- Keeping up to date, rebuilding
- Mixing PPA and official sources

Lime microsystems

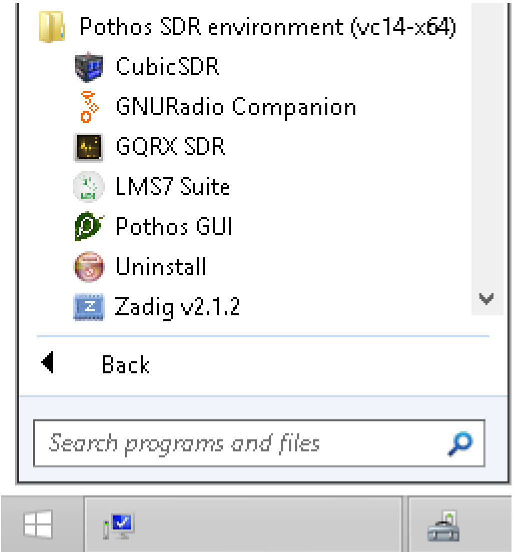
Software packaging: Ubuntu SNAPs

- SNAPs are transactional packages
- Totally contains software stacks
  - Easy to install/remove
  - No DLL/ABI/so hell
- Make a YAML file that tells snapcraft how to build your software stack
  - All dependencies (both from apt-get and source builds)
  - Desired versions/releases of specific software packages
- Get a redistributable installer file that can be installed or shared
  - Or upload the .snap file for distribution through a SNAP store
- Lots of **examples** using LimeSuite and GNU Radio software stacks
  - GUI, command line, and server style examples
  - https://github.com/myriadrf/snapcraft-sandbox/blob/master/README.md

Blog: https://myriadrf.org/blog/snap-packages-limesdr/

Software packaging: Windows - PothosSDR

- PothosSDR is an open source build environment for the SDR ecosystem
  - Homepage: https://github.com/pothosware/PothosSDR/wiki
- SoapySDR, LimeSuite, Pothos, CubicSDR, GRC, GQRX and dependencies….
- CMake project with NSIS and ExternalProject_Add()
  - Nearly 60 software packages, most build from source
- Installer under 80 MB – Post install boost dev, qt dev, or python based on needs



- Integrated: Installer writes registry for Python module paths, environment vars, file extension icon and launcher association
- Custom GRC launcher for sanity checks, automatic module installation, and icon association
  - https://github.com/pothosware/gnuradio-companion-exe
- Getting setup (GNURadio):
  - https://github.com/pothosware/PothosSDR/wiki/Tutorial
  - https://github.com/pothosware/PothosSDR/wiki/GNURadio

Summary

- SDR is built on diverse set of drivers and APIs churning under the hood
- SoapySDR is a cool and versatile tool for the SDR community :-)
- LimeSuite makes it easier to develop applications and hardware based on LMS7002M
- Packaging efforts for the community: PPAs, SNAPs, and Windows installers

Thanks for watching!

Questions/Comments?