MIMOGui

MIMO Graphical Demonstration

REQUIRED PARTS

- **2+ IRIS-030 Software-Defined Radio (SDR) Modules with RF Frontends**
- **4+ Antennas**
- **1x Computer**
- **1x Ethernet Switch with DHCP**
- **3+ Ethernet Cables**

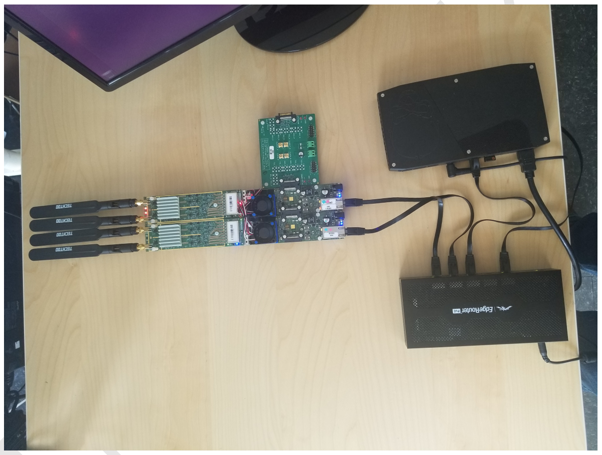

Figure 1: Demonstration setup with PoE power. Test board (top middle) is not required for MIMOGui.

SUMMARY

This guide describes how to set up two or more IRIS-030 Software-Defined Radio (SDR) modules with analog RF front-ends such that their reference clocks are both frequency-matched and phase-locked for coherent beamforming. This guide further provides reference code for using IRIS-030 synchronized frame triggering functions to transmit and receive a coherent MIMO frame from any number of IRIS-030 modules, using a hardware abstraction framework called "SoapySDR." Additional examples of real-time display of transmitted data are also provided.

This guide should serve as a starting point for developers wishing to construct arbitrary MIMO radio topologies and perform controlled experiments using Skylark's IRIS products.

WARNINGS

ALWAYS ensure Iris hardware has heatsinks attached and is properly cooled to avoid damage to the hardware!

Do NOT power on the Iris without attaching an antenna or terminator to the RF outputs. High-power RF amplifiers can be easily damaged by being turned on without proper termination.

Do NOT hot plug Iris modules into each other without first disconnecting from power. Failure to power down Iris modules before changing physical bus configuration can result in permanent damage to the FPGA GPIO pins.

Do NOT increase the transmit gain when using amplified RF front ends without consulting Skylark Wireless or your specific front end documentation. The Iris SDR modules provided intentionally do not lock out potentially unsafe gain settings, since they are required for optimal power output depending on the digital waveform. Increasing gain without monitoring output power levels may damage or destroy the RF power amplifiers.

HARDWARE SETUP

1. Plug antennas in to Iris RF ports.
2. Connect Iris radio modules together as shown in Figure 1.
3. Connect Ethernet cables from the Irises and computer to DHCP enabled switch.
4. Power on Irises, DHCP switch, and computer.

Ensure that the Irises have SD cards installed, and are flashed with the latest software build. See the Iris Quickstart guide for more information.

MIMOGui requires all Irises to be synchronized over their bus connection, unless you are using "LTSMode", described below, which can leave one Iris unsynchronized.

SOFTWARE SETUP

This is a very abbreviated tutorial on setting up the software; for more complete setup instructions see the Iris Quickstart guide. MIMOGui is a Python 3 application that uses SoapySDR and pyqtgraph, along with the following Python modules: numpy, opengl, pyqt, pyqtgraph. MIMOGui.py depends on two external files: LTS.py and logo.tif. SoapySDR and installation instructions can be found on Skylark's GitHub repository:

https://github.com/skylarkwireless/sklk-soapyiris

Linux

To setup SoapySDR and install dependencies, simply run the `install_soapy.sh --extra` script on an Ubuntu 16.04 or later computer. This script may be found within the `sklk-soapyiris/utils/` folder. This script can be easily adapted to other versions of Linux.

Windows

We recommend using WinPython with Windows [1]. Use pip to install the required packages, listed above, if necessary: `pip install XXX` Install the latest build of SoapySDR for Windows from the official website [2].

Check Connectivity

Once the computer is booted, you should be able to open a terminal and run `SoapySDRUtil --find` and see all connected Irises have been discovered, and information about their status.

RUNNING MIMO DEMO

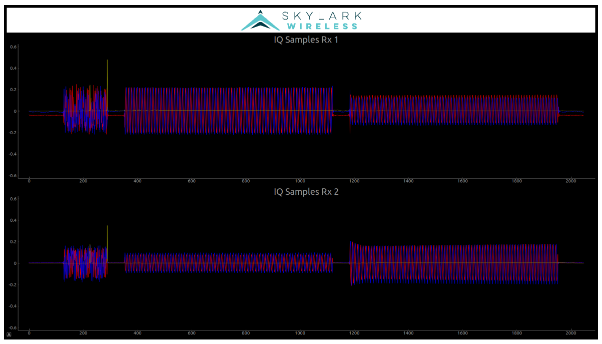

Figure 2: MIMOGui demo in LTSMode showing 802.11 detection and coherent sine waves.

To open MIMOGui, simply run the script using Python 3 specifying the serial numbers of the Irises you are using, e.g., `python3 MIMOGui.py --serials 0123 2345`. This script can be found within the `sklk-soapyiris/demos/` folder. If properly installed, Python 3 can be called from a terminal (cmd.exe on Windows) or an IDE, such as Spyder. **The first serial number argument should be the first Iris in the chain, as it will trigger the other Irises downstream.** This demo sets the first half of the Irises to transmit and the second half to receive. The transmit Irises send non-overlapping sinusoids in time, while the receive Irises listen on both channels and plot the time domain of the received IQ in red and blue on separate graphs.

Note that more Irises can be used to receive by simply adding their serial numbers to the list on the command line.

To exit, simply press ‘q’ (make sure the graphic window is selected). ‘Esc’ will exit full-screen mode, and ’f’ will toggle full-screen. See the “–help” printout for more options.

LTS Mode

An optional “–LTSMode” mode can be enabled to demonstrate un-synchronized transmission, e.g., `python3 MIMOGui.py --LTSMode --serials 0123 2345`. This demo uses a single standalone Iris, the last serial in the list, to repeatedly transmit an 802.11 long training symbol (LTS), then non-overlapping sinusoids on both antennas. All other Irises, which are assumed to be synchronized with their bus connection, listen on both channels and plot the time domain of the received IQ in red and blue on separate graphs. The yellow line is the correlation detection of the LTS preamble, which is used to align the samples in time. Optionally, the “–Constellation” switch can be used to change the signal to a basic OFDM signal and plot the symbols in the frequency domain with very rudimentary equalization.

REFERENCES

[1] “Winpython,” May 2018. http://winpython.github.io/.

[2] “Pothossdr,” May 2018. http://downloads.myriadrf.org/builds/PothosSDR/?C=M;O=D.

DOCUMENT REVISION HISTORY

Date	Revision	Revision Description
2018/05/17	0.8	Initial document creation.
2018/05/28	0.9	Preliminary release.

NOTICE OF DISCLAIMER