Joel Brigida

Transcribed from 2/12/24 Meeting w/ Project advisor Dr. Giorgios Sklivanitis.

1. Collect all 5 PlutoSDRs & bring to lab w/ parts and cables. Connect them all to the common clock source & set them to the same freq. We will need a powered USB Hub.
2. Run a test with the USB hub & check that we can communicate with all PlutoSDRs separately.
   1. This can be accomplished by changing IP addresses or COM ports.
3. Open GNU Radio Companion.
   1. Create 4 different instances of PlutoSDR Source blocks. In these blocks, we can change the carrier frequency, SW gain, sample rate, etc.
4. Activate all the PlutoSDRs on the same frequency for TX. Connect each PlutoSDR output to an oscilloscope channel.
   1. Read the frequency in the time domain and calculate errors. They should all be the same.
5. Using the 5th spare Pluto, set the other 4 to RX and then set this 5th Pluto as a transmitter using GNU Radio.
   1. Use a signal source block, which generates a sinusoid @ the known frequency. That signal source gets connected to a Pluto source block to control the transmitter.
   2. Specify the transmit port being used.
   3. Use a splitter block to split the TX
6. If the transmitters / receivers are synchronized, the oscilloscope output should show this condition.
7. There may be a DELAY from when each I/Q sample comes in to the PlutoSDRs, but in that case, we can use the “delay” blocks to compensate.
8. We can also plot the external clock output on the oscilloscope.

Other Notes:

* The Multi-RTL GNU radio block includes delays to compensate for the USB processing delays.
* There is a tool to generate GNU Radio blocks so we can see what a template block looks like. Dr. Sklivanitis offered to help us with this if we need help.
* We could begin by using 4 different PlutoSDR source blocks and playing around with the delays. We may not be able to avoid asynchronous sample delays received by the PlutoSDR.
* We may be able to “cheat” and use a separate delay block for each Pluto Source block and manually tweak them.

Generating a GNU Radio Block:

Need to build C++ and \*.h Header files for each block. Then we need to build the XML file. This is what enables the block to be a part of GNU Radio Companion so that it is visible.

Every GNU Radio block has a specific structure of folders: Includes, Libraries, Headers, etc....we need to figure this out. (JB Note: Maybe there’s a YouTube tutorial??)

RESOURCES:

1. [[General GNU Radio Block Structure](https://wiki.gnuradio.org/index.php/BlocksCodingGuide)] {https://wiki.gnuradio.org/index.php/BlocksCodingGuide}
2. [[Creating Python OOT with gr-modtool](https://wiki.gnuradio.org/index.php?title=Creating_Python_OOT_with_gr-modtool)] {https://wiki.gnuradio.org/index.php?title=Creating\_Python\_OOT\_with\_gr-modtool}
3. [[Imbedded Python Block Reference](https://wiki.gnuradio.org/index.php?title=Creating_Your_First_Block)] {https://wiki.gnuradio.org/index.php?title=Creating\_Your\_First\_Block}
4. [[GNU Radio Delay Block API Reference](https://www.gnuradio.org/doc/doxygen-3.7.5/classgr_1_1blocks_1_1delay.html)] {https://www.gnuradio.org/doc/doxygen-3.7.5/classgr\_1\_1blocks\_1\_1delay.html}
5. [[GNU Radio Delay Block Implementation](https://github.com/gnuradio/gnuradio/blob/7d61746e27778c56eb8a805ad940c89ff265e313/gr-blocks/lib/delay_impl.cc#L27)] {https://github.com/gnuradio/gnuradio/blob/7d61746e27778c56eb8a805ad940c89ff265e313/gr-blocks/lib/delay\_impl.cc#L27}
6. [[GNU Radio PlutoSDR Source Reference](https://wiki.gnuradio.org/index.php/PlutoSDR_Source)] {https://wiki.gnuradio.org/index.php/PlutoSDR\_Source}
7. [[GNU Radio’s Block-Building Comprehensive Video Tutorial – Pythonic Method](https://youtu.be/CnJObODsx0I?si=VmzfgwWea_bgkDjW)] [1:28:12] {https://youtu.be/CnJObODsx0I?si=VmzfgwWea\_bgkDjW}
8. [[Simple GNU Radio Block Creation Video Tutorial – gr-modtool C++](https://youtu.be/0hJgCRnHTqM?si=CPZr820Cl2J1M_kw)] [15:58] {https://youtu.be/0hJgCRnHTqM?si=CPZr820Cl2J1M\_kw}

Also note that Dr. Sklivanitis recommends we submit an Abstract to the GNU Radio Conference and we may be invited to present our project at the conference. He stated that “The Department” can cover costs for travel to [Tennessee??]

**Using The KrakenSDR for testing**

1. Use the KrakenSDR VM which has the KrakenSDR GNU radio block pre-installed.
2. Set up the Circular array with the 3D printed jig.
3. Receive samples from the KrakenSDR
4. Use a PlutoSDR as a transmitter to test the Kraken MUSIC DoA algorithm and achieve repeatable results.
5. Affix the transmitter and take measurements in relation to the Circular Array (Distance to PlutoSDR source).
6. We can use this setup to test and develop a mathematical model for RX beamforming after taking measurements, since we know the speed of light and the frequency.

If we want to start with a Linear Array, then we can set up all 5 Kraken Antennas in a line spaced at and then take a PlutoSDR and start transmitting. If the transmitter is at 0 degrees, then we know what to expect for delay at the 2nd, 3rd, 4th antennas for delay because we know the distance to the transmitter. This can be used to test and validate our algorithms.

Then we can migrate to a circular array and take all the measurements. Write the model code and test with the Kraken to see if the mathematical model matches.