**BUILD INSTRUCTIONS**

Peyton Adkins – Joel Brigida – Alexandra Calvo – Jason Kurth – Gabriel Faro

**ABSTRACT**

Our project focuses on the concept of phased array beamforming, a primarily software-defined technology that advances traditional RADAR and telecommunications technologies. Our methods have involved the development of custom antennas for testing, the integration off-the-shelf software defined radios modified with other low-cost parts, modification and expansion of existing algorithms, and the development of a graphical user interface for user control.

This file serves as the basis for replicating this project by relaying basic components that make up a fully testable demonstration. Specific instructions are relayed in the `testing.docx` file located in the parent directory whereas general topics and specific hardware and software requirements are discussed here.

**HARDWARE REQUIREMENTS**

Major hardware components include tools for signal processing via antennas, the software-defined radios involved, tools for the connectivity and frequency discipline of the multiple SDRs (software defined radios), and other tools. This project is designed to run on any x86\_64 architecture, laptop, or a desktop.

**MODIFIED PLUTO SDRS**

Our project involved the use of a specific Software Defined Radio named the PlutoSDR, which is manufactured by Analog Devices. [PlutoSDR Specifications](https://www.analog.com/en/resources/evaluation-hardware-and-software/evaluation-boards-kits/adalm-pluto.html#eb-overview)

With only one Tx and one Rx node (antenna) included by default, we had followed the instruction of Analog Devices Inc. Associate *Jon Kraft* for the modification of these SDRs for dual Tx and Rx capabilities. Below are some sources that highlight Kraft’s work:

[YouTube: Jon Kraft, Enable Dual Rx Dual Tx](https://www.youtube.com/watch?v=ph0Kv4SgSuI)

[Python: Test Script to Test Dual Tx and Dual Rx [Jon Kraft]](https://github.com/jonkraft/PlutoSDR_Labs/blob/master/Pluto_revC_rev2.py)

Below is the source of a major beamforming algorithm used in the development of our project:

[YouTube: Jon Kraft, Monopulse Tracking](https://www.youtube.com/watch?v=XP8OWMDHfOQ)

Performing the PlutoSDR Firmware Modifications: This requires a terminal emulator like [PuTTY](https://putty.org/) to log in. Since the PlutoSDR connects with the host PC via IP address, we can open PuTTY and enter the IP address to log in. We need to change some firmware settings for two specific circumstances: Dual transmit and Dual receive channels and accepting an external 40MHz clock source. This process, with screenshots, is written about in detail with screenshots in the ‘doc -> research’ folder. The filename is ‘Joel\_README\_PlutoSDR-Research.pdf’

After the firmware modifications are completed, we need to change the PlutoSDRs from their original IP address of 192.168.2.1. This must be done one at a time, or an IP conflict will happen if more than one PlutoSDR is plugged in with the same IP Address. In that case, they should both be unplugged and then only plugged in one at a time.

To do this, we need to access the PlutoSDR itself. The Icon is on the desktop in the upper left. Double click it to see the contents. In that folder, there is a file called ‘config.txt’. Open that file. Look for the section that says:

[NETWORK]

hostname = pluto

ipaddr = 192.168.2.1

ipaddr\_host = 192.168.2.10

netmask = 255.255.255.0

Since we have 5 PlutoSDRs to set up, we need to configure all 5 in the following order before they can all be plugged into the same PC.

Change the ‘ipaddr’ and ‘ipaddr\_host’ for each Pluto:

Pluto 1: ipaddr = 192.168.2.1, ipaddr\_host = 192.168.2.10

Pluto 2: ipaddr = 192.168.3.1, ipaddr\_host = 192.168.3.10

Pluto 3: ipaddr = 192.168.4.1, ipaddr\_host = 192.168.4.10

Pluto 4: ipaddr = 192.168.5.1, ipaddr\_host = 192.168.5.10

Pluto 5: ipaddr = 192.168.6.1, ipaddr\_host = 192.168.6.10

Note that after the contents of the ‘config.txt’ file is changed, the PlutoSDR device should be ejected, then wait until it reconnects again. The new ‘config.txt’ should reflect the IP change. Now the PlutoSDR device can be unplugged.

Once all IP addresses are changed like above, all the PlutoSDRs can be plugged into the generic USB 3.0 powered hub. This hub then plugs into the computer. At this point all 5 PlutoSDRs should be connected and shown on the desktop. We are now ready to start making antenna connections with coaxial cable.

For direction finding with the UCA, one RX port from each PlutoSDR is connected to one coaxial cable, leading to one dipole. This setup can now track an unknown transmitter from any direction around the array. This 5 element UCA is also made to function with the KrakenSDR, which has 5 channels. This is why this array has 5 elements. When using a ULA for direction finding, both RX ports from 4 of the PlutoSDRs for a total of 8 elements, but keep in mind the ambiguity problem of the linear array.

**EXTERNAL CLOCK SOURCE**

The base for the external master clock device is the CDCLVC1310-EVM. It needs to be modified to run off a 40 MHz crystal. Using hot air, remove the on-board 25 MHz crystal and replace it with a 40 MHz version, part # [ABLS2-40.000MHZ-D4YF-T](https://www.mouser.com/ProductDetail/ABRACON/ABLS2-40.000MHZ-D4YF-T?qs=LoTOQoUkC8SsxxUxj0Db0w%3D%3D).

The VDD and VDDO levels are adjustable and set by a [Buck Converter DC/DC 3v-12v to 1v-5v Volt Module](https://www.amazon.com/dp/B0BHDNWH59?ref=ppx_yo2ov_dt_b_product_details&th=1), with a 12v DC input and adjusted to provide 2.5v output and power the VDD and VDDO inputs on the TI module. The module has banana jacks for attachment to a lab power supply, which were removed and then the outputs directly wired and soldered to the board.

The Pluto SDRs were removed from their plastic housings. M3 hex standoffs were used to stack them and attach them to the acrylic platform. 12mm high standoffs were used with M3 screws. [6" u.FL to SMA cables](https://www.amazon.com/HiLetgo-Wireless-Antenna-Extension-NRF24L01/dp/B01HXU1PKS/) were used to attach to the clock in port on the Pluto boards, and then an [SMA-Male to SMA-Male adapter](https://www.amazon.com/VCE-3-Pack-Coaxial-Adapter-Connector/dp/B06Y64DS9V/) was used to attach to the TI External Clock board at each clock out port.

A generic USB 3 powered

4-port USB hub was also added to the board to attach the PlutoSDRs over USB, and a second Buck-Boost converter was used to step down to 5v to power the hub from the same 12v DC brick. A piece of strip board was used to wire the 12v sources to the different converters and add a DC pigtail to attach the external 12v DC power brick.

**ACRYLIC BASE**

The acrylic base was designed in CAD (Computer Aided Drafting) and then the file was laser cut out of ¼" glossy acrylic with all mounting points. Vinyl stick-on feet were used for the bottom and M3 pan head screws were used to attach all M3 standoffs. The files are attached.

**ANTENNA EQUIPMENT**

The equipment used for both antenna arrays in this project is shown in the bulleted list below. Most of the antenna equipment was shared as a means of saving money and time, since many of the same concepts are utilized in both arrays (like the base and mast).

Assorted PVC Pipes and fittings:

* 1-½” PVC Pipe
* 1” PVC Pipe
* [3-way PVC splitter](https://www.amazon.com/dp/B08HJ2FZL9?psc=1&ref=ppx_yo2ov_dt_b_product_details)
* [1-½” to 1” 3-way PVC splitter converter](https://www.amazon.com/dp/B004UHF1VM?psc=1&ref=ppx_yo2ov_dt_b_product_details)

Adjustable PVC Arms:

* 1” PVC pipe
* 5 brushed aluminum stand-off screws
* [915 MHz pre-made Dipole antennas.](https://lowpowerlab.com/shop/product/193)

3D Printed Custom Parts (files attached):

* 6-way splitter
* Caps to align Dipoles
* Base with leg extensions
* Antenna arms for linear array
* [3D printer filament](https://www.amazon.com/dp/B014VM9724?psc=1&ref=ppx_yo2ov_dt_b_product_details)

Coax Cables:

* [6 ft. SMA-M to SMA-M RG-58/U coax](https://www.amazon.com/gp/product/B07T162T9J/ref=ox_sc_act_title_1?smid=A1THAZDOWP300U&th=1)

Rubber O Rings

* [Rubber O-Ring Kit](https://www.amazon.com/dp/B0BRN1H471?psc=1&ref=ppx_yo2ov_dt_b_product_details)

SMA Adapter

* [SMA 90 Degree Adapter (x8)](https://www.amazon.com/Antenna-Connector-RFAdapter-Fatshark-Transmitter/dp/B07XHJQDBT/)

**CUSTOM ANTENNA SPECIFICATIONS**

Circular Antenna Array:

The circular array was originally made with the idea of using PVC pipes for the whole thing. Although it was very cost-effective to use PVC pipes, it was found later that it could not be used for everything we needed. However, we did find that 3D printing custom parts for the antenna array would be much better. The circular array was made so that it could be fully disassembled and re-assembled, especially since we already had the idea of using PVC pipes to begin with. So, the outcome of this is a mixture of PVC pipes and 3D printing to bring the antenna design to life.

When it comes to re-creating the antenna array it is as simple as reprinting the provided files for the 3D printed parts and putting them together with a 1.5” PVC pipe, as those parts were made for that. If the individual re-creating the antenna wants to downscale it, it is doable with the set up that is there, especially since the whole project was done with the idea of it being updateable. The reason we did not downscale past the 1.5” PVC was that we wanted to hide the coax cables inside of the PVC, and the more elements that are added means that more space is needed to hide all the cables together. This design was made with the idea that there would not be more than 10 elements, as 10 cables would be tight inside of the 1.5” PVC. However, all of this is adaptable, as the base 3D print has a hole that is exposed all the way at the bottom, so modifications can be made to accommodate more elements. This antenna array was also made with the idea of inter-element spacing in mind, which is where the 6-way splitter on top of the circular array comes into play. There are screws that make it so that the antenna arms are adjustable, giving some room to set the inter-element spacing correctly (half a wavelength) for each element.

Putting the array together is simple, as far as the center mast, base, and top parts go. There is a slightly tricky part when it comes to putting the wires through the arms, connected to the antenna element, into the center mast to come out at the exit hole at the base in the bottom. The tricky part is that you cannot bend the wires too much, and by the time the last antenna is being connected, it is more difficult to pull the wire through, so one needs to be careful when setting it up for the first time. Once set up though, the wires don't have to be removed until the entire antenna array is needed to be disassembled. Operating the antenna array is also very simple once it is all put together. The top part has 5 screws and 5 arms, as described earlier, the screws are for adjusting the arms, moving them either close in or further out, depending on the adjustment for the antenna element spacing. We have our antenna arm lengths set to be fine-tuned at half a wavelength, so that the arm does not have to be unnecessarily long.

Linear Antenna Array:

The linear antenna array initial idea was to be built out of PVC pipes for the whole design. While this design was technically good it did not fit our needs. What I did next was design a 3D model of our next arm for the antennas. To create these 3D models, I used the program Blender [Download Blender](https://www.blender.org/download/) first but then found out that the program Fusion 360 [Download Fusion 360](https://www.autodesk.com/products/fusion-360/free-trial?mktvar002=5022374|SEM|9470270545|99799579647|kwd-296858593645&utm_source=GGL&utm_medium=SEM&utm_campaign=GGL_D-M_Fusion-360_AMER_US_Trials_SEM_BR_NEW_EX_0000_5022374&utm_id=5022374&utm_term=kwd-296858593645&mkwid=s|pcrid|647627026484|pkw|download%20fusion%20360|pmt|e|pdv|c|slid||pgrid|99799579647|ptaid|kwd-296858593645|pid|&utm_medium=cpc&utm_source=google&utm_campaign=GGL_DM_Fusion-360_AMER_US_Trials_SEM_BR_NEW_EX_ADSK_3529781_&utm_term=download%20fusion%20360&utm_content=s|pcrid|647627026484|pkw|download%20fusion%20360|pmt|e|pdv|c|slid||pgrid|99799579647|ptaid|kwd-296858593645|&gad_source=1&gclid=Cj0KCQjw8pKxBhD_ARIsAPrG45kDycDfr1MqIYhzJDdUhABCaCSJuvBLG5Fe-yBeKMdA-CvYJGjGVt4aAqKLEALw_wcB&ef_id=ZQuBCwAR0bNlAwAN:20240421234233:s) doing a better job designing the arms. The way we thought of doing the arms was to instead make an exact hole for the dipoles but then we thought to make it like a slider so that way we could adjust at any inter-element spacing.

When building the linear antenna array you would need 2 PVC 1” x 1” x 1” fittings as well as a 1” x 1-½” x 1” fitting. Also, 2 PVC pipes to attach to the main fitting to the other 2 fittings. From those two fittings we will add 3D printed arms on each side of the fitting, having 4 arms in total. On each arm we will have 2- 915 MHz dipole antennas (8 antennas in total for the array) then those antennas go on the slider hole and on the other side we would put one of the rubber O ring (the size used in this kit is the 5x2mm), this rubber O ring would help the dipole antenna to stay in place. Then attach the SMA 90-degree adapter to the connector of the dipole, on the other end of adapter we are going to connect the Coax cable one per antenna. Label each cable to know which cable is which since they will be going through the 3D printed arms and the PVC pipes. The main fitting that is an adapter as well 1” x 1-½” x 1” will be connecting both linear arrays together and there will be a 1-½" PVC pipe attached to it. Then, the last part is a 3D printed base that has a hole to pass all the coax cables through. All the 3D printed parts have their own files so they can be recreated at any point.

**SOFTWARE REQUIREMENTS**

Software for our project is primarily written in Python, making use of the Python libraries for the Pluto SDR (software defined radios) via Analog Devices. Some experimental development and testing have been conducted using software tools such as GNU Radio.

**DEVELOPMENT ENVIRONMENT**

To build this project in its native environment, some care and expertise must be taken when using Partitioning software like GParted, and installing Operating Systems like Linux Mint. The safest way to do so is to use a brand-new hard drive to avoid losing data on a drive which is already in use. This was not an option in our case. If you do not already know how to install an Operating System and/or partition hard drives, then you should not attempt this without someone who is familiar with these procedures. This guide does not serve as a general guide to Operating Systems Installations or Partitioning software.

This project was built in an open-source Linux Mint 21.3 OS that can be downloaded free here: [Download Linux Mint 21.3.](https://www.linuxmint.com/download.php) Care must be taken when installing a new Operating System! Data Loss can result from small mistakes! Do not attempt this unless you have successfully done this before.

To partition a hard drive which is already in use as I did, the open-source tool GParted, which stands for Gnome Partition Editor, can be downloaded for free here: [GParted - Download](https://gparted.org/download.php)

Note that this step is performed as needed. A fresh hard drive would not require partitioning, the OS installation would do it automatically.

One of the tools used for testing our phase coherence was the KrakenSDR, which can be researched and/or bought here: [Home | KrakenRF](https://www.krakenrf.com/)

This software installation will install additional dependencies to use the KrakenSDR as well as run our project experiment. If you do not wish to use a KrakenSDR, this software installation will not affect anything else.

1. Perform operating system installation of Linux Mint 21.3.
   1. On a new hard drive: no partitioning needed
   2. To run dual-boot or something custom: hard drive partitioning is required as the user sees fit.
2. After the Linux Mint 21.3 operating system is installed, all the system updates must be installed. Click the “Menu” on the lower left of the desktop and type in the search box “update” and select the “Update Manager” in the list. Run the update manager and install all the updates it suggests. Reboot the computer after this process since there will be many updates selected. The operating system is now up to date. Next, we will install project specific software.

**GNU Radio and KrakenSDR SW Install:**

1. Install the following per the [KrakenSDR Install Script](https://github.com/krakenrf/krakensdr_docs/wiki/09.-VirtualBox,-Docker-Images-and-Install-Scripts#install-scripts) for x86\_64 by opening a Terminal and entering the following at the default bash prompt:

wget https://raw.githubusercontent.com/krakenrf/krakensdr\_docs/main/install\_scripts/krakensdr\_x86\_install\_doa.sh

sudo chmod +x krakensdr\_x86\_install\_doa.sh

./krakensdr\_x86\_install\_doa.sh

This script will take a few minutes to install, so be patient. After it finishes, reboot again.

1. Open a terminal on the desktop after rebooting and enter the following:

sudo apt update

sudo apt-get install g++-12

1. When finished, install the following per the [Kraken Forum Recommendation](https://forum.krakenrf.com/t/issue-while-installing-using-scripts-x86-64bit-system/1088/4)

sudo apt update

sudo apt-get install libc++-dev libc++abi-dev

1. Install GNU Radio and PPA:

sudo add-apt-repository ppa:gnuradio/gnuradio-releases

sudo apt-get update

sudo apt-get install gnuradio python3-packaging

1. Make a new directory, then download and run the [Install Scripts:](https://github.com/krakenrf/krakensdr_docs/wiki/09.-VirtualBox,-Docker-Images-and-Install-Scripts#install-scripts)

mkdir ~/krakensdr\_doa

cd krakensdr\_doa

wget https://raw.githubusercontent.com/krakenrf/krakensdr\_docs/main/install\_scripts/krakensdr\_x86\_install\_doa.sh

sudo chmod +x krakensdr\_x86\_install\_doa.sh

./krakensdr\_x86\_install\_doa.sh

1. Note that during this step, the ’make’ process fails due to a missing ’rtl-sdr.h’ file not found. Here is a snippet of the output:

Cloning into 'heimdall\_daq\_fw'...

remote: Enumerating objects: 1398, done.

remote: Counting objects: 100% (270/270), done.

remote: Compressing objects: 100% (44/44), done.

remote: Total 1398 (delta 241), reused 230 (delta 226), pack-reused 1128

Receiving objects: 100% (1398/1398), 1.95 MiB | 4.95 MiB/s, done.

Resolving deltas: 100% (910/910), done.

cp: cannot stat '/home/sdr/librtlsdr/build/src/librtlsdr.a': No such file or directory

cp: cannot stat '/home/sdr/librtlsdr/include/rtl-sdr.h': No such file or directory

cp: cannot stat '/home/sdr/librtlsdr/include/rtl-sdr\_export.h': No such file or directory

gcc -Wall -std=gnu99 -march=native -O2 -I. -c -o ini.o ini.c

gcc -Wall -std=gnu99 -march=native -O2 -I. -c -o log.o log.c

gcc -Wall -std=gnu99 -march=native -O2 -I. -c -o iq\_header.o iq\_header.c

gcc -Wall -std=gnu99 -march=native -O2 -I. -c -o sh\_mem\_util.o sh\_mem\_util.c

gcc -Wall -std=gnu99 -march=native -O2 -I. log.o ini.o iq\_header.o -o rtl\_daq.out rtl\_daq.c -lpthread -lzmq -L. -lrtlsdr -lusb-1.0

rtl\_daq.c:49:10: fatal error: rtl-sdr.h: No such file or directory

49 | #include "rtl-sdr.h"

| ^~~~~~~~~~~

compilation terminated.

1. To fix the above error, delete and reinstall the ‘librtlsdr’ library:

sudo apt purge librtlsdr\*

sudo rm -rf /usr/lib/librtlsdr\* /usr/include/rtl-sdr\*

/usr/local/lib/librtlsdr\* /usr/local/include/rtl-sdr\*

1. Reinstall ‘librtlsdr’ to system:

git clone https://github.com/krakenrf/librtlsdr

cd librtlsdr

mkdir build

cd build

cmake ../ -DINSTALL\_UDEV\_RULES=ON

make

sudo make install

sudo ldconfig echo 'blacklist dvb\_usb\_rtl28xxu' | sudo tee –append /etc/modprobe.d/blacklist-dvb\_usb\_rtl28xxu.conf

1. Remake the ‘\_daq\_core’ and now it will build:

cd ~/krakensdr\_doa/heimdall\_daq\_fw/Firmware/\_daq\_core

make

1. Install ‘gr-krakensdr' GNU Radio Block as per [GNU Radio OOT Block Install Instructions](https://github.com/krakenrf/gr-krakensdr)

sudo apt-get install gnuradio-dev cmake libspdlog-dev clang-format

git clone https://github.com/krakenrf/gr-krakensdr

cd gr-krakensdr

mkdir build

cd build

cmake ..

make

sudo make install

1. Edit ‘daq\_chain\_config.ini’:

cd ~/krakensdr\_doa/heimdall\_daq\_fw/Firmware

nano daq\_chain\_config.ini

Make the following changes:

[data\_interface]

out\_data\_iface\_type = eth

1. The system is now ready to operate with GNU Radio and/or the KrakenSDR. If you wish to use the KrakenSDR with the desktop, then open a terminal and enter the following:

cd krakensdr\_doa

./heimdall\_only\_start.sh

1. This script starts and shows ‘Have Coffee Watch Radar’ in the terminal to indicate that it is working properly.
2. Open a new terminal and run ‘gnuradio-companion &’
3. Try to open and run the in the ‘kraken\_fft\_display.grc’ in the ‘~/gr-krakensdr/examples’ folder. This script should start and run with the KrakenSDR connected and powered on. There should be a GUI window with the FFT display for each of the 5 channels. This indicates that both GNU Radio and the KrakenSDR are working properly.

Everything required for GNU Radio and KrakenSDR has now been installed. These systems can now be tested for proper operation.

**DEPENDENCIES, LIBRARIES**

To run the python code for beamforming and direction-finding, which is our code for the presentation, we need to install additional libraries and Visual Studio Code

1. Install Visual Studio Code Editor. [Download Here](https://code.visualstudio.com/download) and then install the package by double clicking the package in its folder and clicking “Install.”
   1. After Installing, click on the “Extensions” icon on the left side bar and search for “python” by Microsoft. Install that extension, which will install a whole bunch of other python extensions.
2. Open a new terminal and create a new python environment using the Micro Mamba that was installed during the KrakenSDR script install:

conda create --name sdr

conda activate sdr

1. Now that a new Python Environment is installed, we can start adding libraries.
   1. We need to add the following packages using the following syntax:

conda install conda-forge::<packageName>

* 1. Install the following packages:

math, numpy, pandas, pylab, pyqtgraph, gnuradio-core, pyqt5, time, pyadi-iio

* 1. Now that the packages are installed, we can deactivate the python environment since we do not need it anymore, then close the terminal:

conda deactivate

All dependencies and libraries have now been installed. VS Code can now run our custom Python source code that is in the ‘src’ folder.

**GITHUB REPOSITORY**

During our project's development, we stored documents and source code in a primary GitHub repository and GUI-specific developments in a connecting repository.

[Primary Project GitHub Repository](https://github.com/ADolbyB/sdr-beamforming)

[GitHub Repo for GUI](https://github.com/RayzrReptile/PyQT_GUI)

The primary repository holds extensive information and links concerning our project, rendering it quite useful as an information center on its own. The README.md files from the project repository have been exported to PDF format and placed in the “Research” folder. Other repositories noted also within the project’s primary repository include:

* + [coherent-receiver's N-Channel Coherent Transceivers](https://coherent-receiver.com/pluto-sdr) (Concept Only).
    - Note that we contacted this company, and they did not offer any information about this product. This led us to believe it was either discontinued or did not work as advertised.
  + [Jon Kraft's Pluto\_Beamformer Repo for PlutoSDR.](https://github.com/jonkraft/Pluto_Beamformer)
  + [Jon Kraft's PlutoSDR\_Labs Repo for PlutoSDR.](https://github.com/jonkraft/PlutoSDR_Labs)
  + [Jon Kraft's PhasedArray Repo for the Analog Devices ADAR-1000.](https://github.com/jonkraft/PhasedArray)
  + [KrakenRF's krakensdr\_doa Repo for Direction Finding on hardware w/ the RPi 4/5 (or on x64 hardware using VirtualBox 7.0+).](https://github.com/krakenrf/krakensdr_doa)
  + [mfkiwl's kraken\_pr Repo for Passive Radar.](https://github.com/mfkiwl/krakensdr_pr)
  + [osmocom's rtl-sdr Repo for The Osmocom RTL-SDR.](https://github.com/osmocom/rtl-sdr)
  + [<https://github.com/ptrkrysik/multi-rtl>] ptrkrysik's multi-rtl Repo for Synchronizing RTL-SDRs.
  + [<https://github.com/ptrkrysik/gr-gsm/tags>] ptrkrysik's gr-gsm Repo for GSM signals on RTL-SDRs.
  + [<https://github.com/analogdevicesinc/gr-iio>] analogdevicesinc's gr-iio Repo for IIO (PlutoSDR) Devices.
  + [<https://github.com/gnuradio/gnuradio>] gnuradio's gnuradio Repo for the GNURadio Program.