**TESTING 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 provides instructions for replicating some basic tests conducted as a preliminary introduction to the concept of beamforming via Jon Kraft and the two primary tests conducted for demonstrative purposes. Hardware and software requirements are discussed in detail within the file `build.docx`

**PRELIMINARY TESTS – JON KRAFT**

Some introductory tests can be conducted after basic hardware and software requirements are met. These tests are developed and documented by Analogue Devices Inc. Associate Jon Kraft. His work gives focus to the concept of phased array beamforming utilizing a single Pluto modified to conduct dual Tx and Rx.

A complete list of videos detailing these tests can be found at his YouTube channel:

<https://www.youtube.com/channel/UCDIF0ZdNn4L45OugpQr75FQ>

A simple test to demonstrate beamforming can be found with his Plot FFT algorithm where he shows the ability to ‘track’ where an incoming signal is sourced from.

<https://www.youtube.com/watch?v=2QXKuEYR4Bw&t=1460s>

The most significant test conducted by Jon Kraft is his Monopulse Tracking algorithm test where he makes use of an algorithm for direction-finding (DoA – direction of arrival). In this algorithm, he demonstrates tracking via the continuous locality of an incoming signal, unlike the Plot FFT algorithm where which continuous locality is not achieved in the same manner.

<https://www.youtube.com/watch?v=XP8OWMDHfOQ>

**BEAM STEERING WITH USER CONTROL**

As a continuation of Jon Kraft’s work, one major component to our project was the ability to control an Rx and a Tx beam. Unfortunately, due to time, we were only able to demonstrate beam steering control on a single Pluto with an Rx beam, similarly to the limitations of Jon Kraft (more information located under `research`). Nevertheless, it is a critical component to the proposed original purpose of the project and allows for a clear demonstration of beamforming.

**HARDWARE SETUP**

The physical setup for this test requires the following:

1. A modified Pluto SDR
2. Three coax cables (two must be of equal length).
3. Three dipole antennas (or, alternatively, logarithmic antennas)
4. Two low-pass filters
5. A PC with the correctly installed dependencies, libraries, and dev environment

Specifications for the individual components can be found within the file `build.docx` to ensure correctness.

**TEST INSTRUCTIONS**

**Step 1**

Find an open area suitable for testing. For best results, find an area outside, being sure to use a table or designated tripods to hold up the antenna. Depending on the frequency of the transmitted signal, you are required to place two receiving nodes (composed of an antenna connected via coax to the two Rx ports of the modified Pluto) apart from one another by *d* distance.

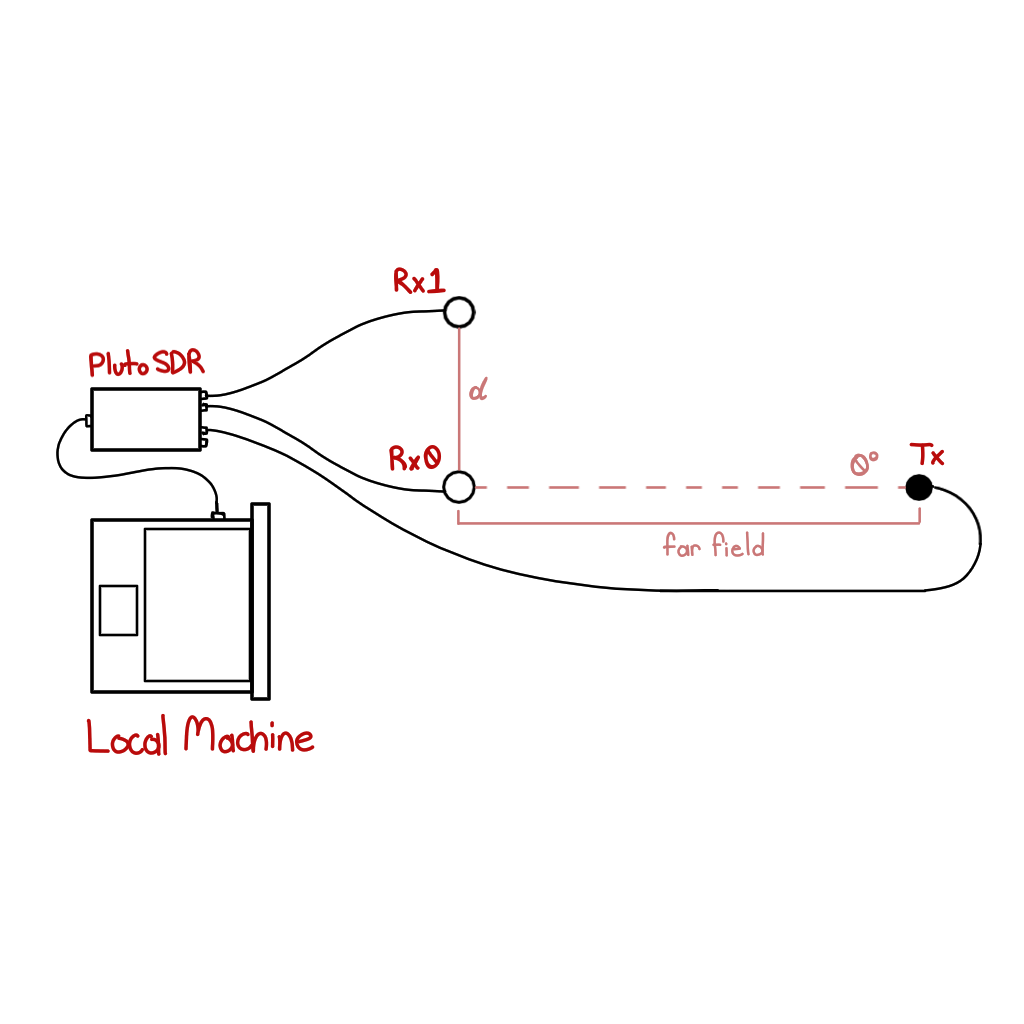
To find this distance, you may consult an online tool to find the half wavelength spacing of your specified frequency or start the DemoBeamSteer\_GUI.py algorithm to see the displayed *d* distance, assuming you can correctly connect to your Pluto.

**Step 2**

After having placed your Rx nodes *d* distance from one another, place your Tx node connected to the original Tx port (Tx0) on your Pluto directly perpendicular to the first Rx node (Rx0). The distance between the Tx node and the array (composed of the two Rx nodes) must be larger than the far field as specified by the frequency of operation.

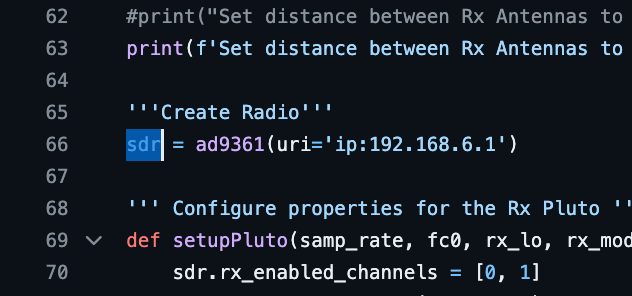
To find this distance, you may consult an online tool.

Your physical setup should now look like the following diagram:



**Step 3**

With your antennas properly placed, you can now connect your Pluto via USB to your local machine set to begin running the script. After confirming that you have successfully connected your Pluto, ensuring that the IP address is the same as the one specified in the script (see below), you can now begin the algorithm.



**Step 4 Onward**

With proper connection, you should be presented with a GUI (Graphical User Interface) window that shows some graphs and various control modules. Details of this GUI are relayed below in the following section.

To terminate the steering process, you must properly close the GUI window; terminating the process traditionally in your IDE will not discontinue the algorithm.

**CONTROLLING THE RX BEAM**

The GUI shows two graphs, some LCD screens, knobs, and buttons. Below is a diagram of the components to the GUI followed by some detailed explanations for major components.

A screenshot of a computer

Description automatically generated

**Pause/Play Button**

The large button in the bottom middle-left controls the steering all together. If in the state that the algorithm is paused, the beam’s direction is completely frozen, though the beam itself does not cease to exist. In the paused state, all other control modules are disabled except for the calibration button.

**Recalibration Button**

The most important component to the functionality of the algorithm, this button is only accessible with the algorithm is paused. To achieve proper phase calibration, the Tx node must be set perpendicular to the first Rx node (Rx0) at the time the phase calibration button is pressed.

This is due to the Rx nodes being slightly off in phase, needing a reference point to properly find a phase offset to continuously use for correction. This reference point exists when there are no additional phase delay values that would add ambiguity to the phase offset used in calibration. As such, an incoming signal at 0º would yield a 0º phase delay, allowing for a more accurate calibration. In essence, the algorithm assigns the direction of the Tx beam as 0º the moment it is pressed.

**The FFT and RADAR Plots/Graphs**

The FFT plot shows the Fast-Fourier Transform display of the summated signals of Rx0 and Rx1relative to the frequency domain. As the Rx beam points more towards the DoA (direction of arrival) of the oncoming signal, the FFT plot should display a visible peak. The peak, shown in blue, is configured to linger momentarily before refreshing upon a cycle, both of which may be toggled.

The RADAR graph shows the direction of the beam with the peak signal direction shown by a blue arrow which, similarly to the FFT graph, will refresh per cycle by default unless otherwise toggled. With your linear array of two nodes, the best reception is estimated to be around 60º on either side, though graphically and computationally, the beam checks for these directions regardless.

**The LCD Screens**

The six LCD screens show the signal strength, phase delay, and estimated DoA for both the continuously moving beam as well as the peak sourced signal when found. If peak display is disabled, the peak LCD values will default to 0.

**The Control Knobs**

The two knobs control both steering speed and phase increment respectively. The beam's speed is more self-explanatory, but the phase increment can also be used to mildly control the speed (by controlling the steps in phase delays tested) and the direction of the beam (by controlling the polarity of the phase). With both knobs, the user can control the speed, direction, and precision of the beam.

**Display Toggles**

The two buttons farthest to the right of the GUI control whether the plot peaks for the RADAR and FFT graphs are displayed and whether they are refreshed upon a new cycle. This is most helpful if one wishes to only display the immediately found signal direction and strength or if a found signal wishes to remain displayed.

**PHASE COHERENCE DEMONSTRATION**

Another key aspect of this project is the complete synchronization of multiple individual Pluto SDR devices, which proved to be the most difficult and time-consuming part of the project. Our group achieved synchronization in both areas of specification: frequency lock and phase lock. Frequency lock required the use of an external clock source for disciplining each of the Pluto SDRs while phase lock required post-process manipulation of received data.

**HARDWARE SETUP**

It is strongly advised to look into the modification specifications in the file `build.docx` for how to construct the hardware for a multi-Pluto system. The following test assumes that this apparatus is constructed, and all necessary parts are available.

The physical setup for this test requires the following:

1. Five modified Pluto SDRs disciplined to an external clock source
2. Five coax cables (Fournmust be of equal length).
3. Five dipole antennas (or, alternatively, logarithmic antennas)
4. Four low-pass filters
5. A five-or-more channel USB bus
6. A PC with the correctly installed dependencies, libraries, and dev environment

**TEST INSTRUCTIONS**

**Step 1**

Find an open area suitable for testing. For best results, find an area outside, being sure to use a table or designated tripods to hold up the antenna. Depending on the frequency of the transmitted signal, you are required to place four receiving nodes apart from one another by *d* distance. These are connected via coax to the first Rx ports (Rx0) of each of the modified Pluto SDRs that are not Pluto1.

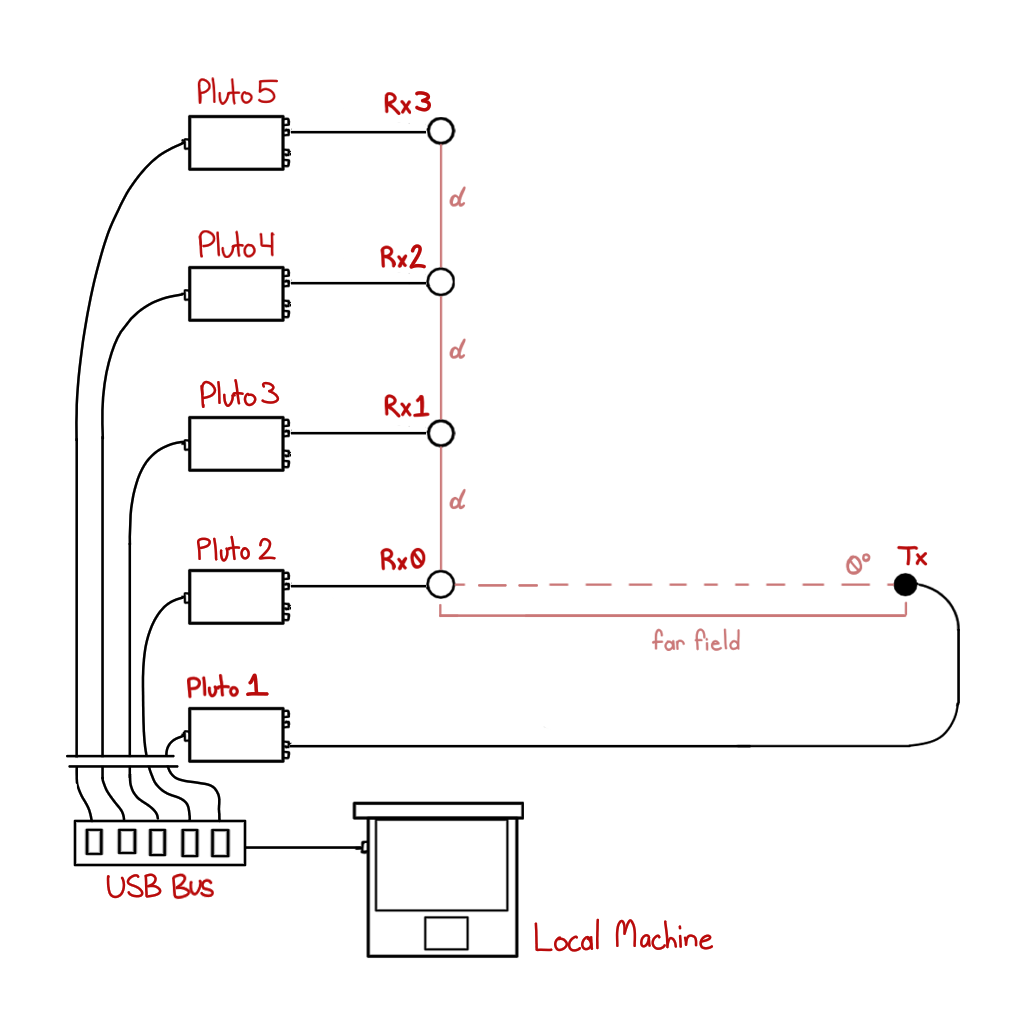
To find this distance, you may consult an online tool to find the half wavelength spacing of your specified frequency or start the `finalDemoPlotPeaks.py` algorithm to see the displayed *d* distance, assuming you can correctly connect to your Pluto SDRs.

**Step 2**

After having placed your Rx nodes *d* distance from one another, place your Tx node connected to the original Tx port (Tx0) on Pluto1 directly perpendicular to the first Rx node (Rx0 of Pluto2). The distance between the Tx node and the array (composed of the four Rx nodes) must be larger than the far field as specified by the frequency of operation.

To find this distance, you may consult an online tool.

Your physical setup should now look like the following diagram:



**Step 3 Onward**

Once your setup is complete with each Pluto connected to your local machine via the USB bus, you may run the program `finalDemoPlotPeaks.py`. Two GUI windows should appear, one displaying the summation of every Rx node along with the estimate DoA as well as a graph depicting the phases of each Rx node in the time domain.

A black background with white text

Description automatically generated

A screenshot of a computer

Description automatically generated

**Summation Graph**

The graph with the large red line shows the strength of the sum of each Rx node’s received signal converted to dbfs, exactly like the plot peaks algorithms from that of Jon Kraft. Here, the red line shows the peak of the summated signals, expected to land consistently near the 0º mark on the x axis. As with a linear array, there exists a problem with *ambiguity*, resulting in some peaks additionally near +180º and –180º. The algorithm, however, limits the range of the phase delays to exclude these values to ensure an expected ≈0º consistency.

**Time Domain Display**

The graph with the jagged peaks ranged between four colors displays the phase values concurrently with each of the Rx nodes. If this display looks more congested than the above image, right-click the GUI to show an option window. Here, lower the percentage range of the x axis to around 1%.

The clearest results of complete phase alignment will occur precisely when the estimated DoA of the oncoming signal is 0º. Otherwise as the summation shifts, the phase offsets will appear misaligned.

**IMPORTANT NOTE FOR THIS TEST**

Due to a lack of time and occasional ventures in other aspects of our project, our implementation of phase coherence is not practical in much of any application. This is due to the strict limitations of the conditions for achieving phase coherence as conducted by our team.

*Phase coherence in this test is only achieved when the known transmitted signal is directly perpendicular to the first Rx node (the reference node).*

Similarly to that of calibrating the beam steering algorithm, a correction for phase offsets or trigger delays between Pluto systems requires the reference node (Rx0) to receive an incoming signal without the required use of a phase shift that would steer the beam. This is to avoid mathematical ambiguity of a value found when calculating the phase offset of a given Rx node. Otherwise, the algorithm would similarly assume that the incoming signal, even if not so, rests at 0º to the reference node.