Boston University
Electrical & Computer Engineering
EC 464 Senior Design Project

Second Prototype Testing Report
3/18/2022

RF Spectrum Analyzer

By
Team 26

Isaac Yamnitsky isaacy@bu.edu
Kakit Wong kakit@bu.edu
Yana Galina yanag@bu.edu
Jaime Mohedano jaimem@bu.edu

Overview

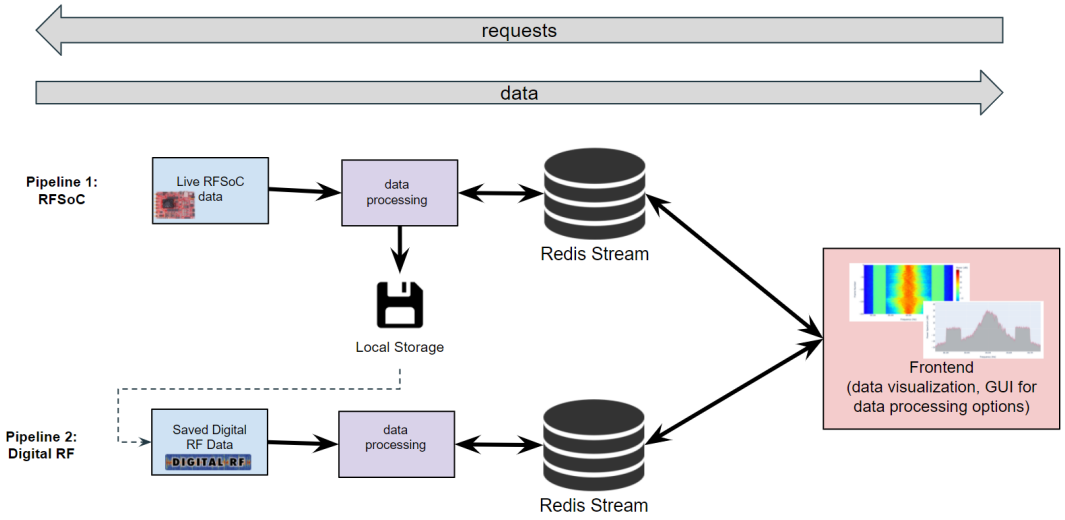

Figure 1: A block diagram of our design

Our team has developed and tested a web application which can display spectrum data for two different pipelines of RF data ("Pipeline 1" and "Pipeline 2", respectively). Pipeline 1 processes live data incoming from a Xilinx RFSoC board, and Pipeline 2 processes locally-stored RF data in the Digital RF format. We tested the second pipeline during our first prototype testing, so for this test we focused on the first pipeline. Our web application displays the RF data with two interactive formats: a spectrum plot and a waterfall spectrogram plot. Overall, we were able to successfully test both pipelines, though some bug fixes and improvements remain to be implemented.

Equipment and Setup

<u>Pipeline 1:</u>
Our setup involves two main components:

- The hardware side consists of the RFSoC board loaded with a data collection Jupyter notebook script. The board is connected to power and the internet via ethernet cable. One RX channel is fitted with an antenna. The other RX channel is connected to one of the TX channels, in "loopback" mode. One computer (PC1) is needed to connect to the board and run the necessary scripts on the board. When the board is powered on and running, the script takes incoming data, converts it to spectrum data using Fast Fourier Transforms, and sends that data over ethernet into a Redis database running on a second computer (PC2).
- The software side consists of a PC2 running the Dash web application as well as the Redis database. This web application processes data streaming

into the Redis database from the board and displays it in the browser with an interactive graph format.

Pipeline 2:

This setup consists of a PC running a redis database, a back end python server, and a front end Dash web app. The back end streams requested Digital RF data into the front end to be displayed on the graphs there, through the Redis database.

Detailed Measurements

Pipeline 1, Trial 1: Antenna

The RFSoC RX0 channel is connected to an antenna. The data from this RX channel is streamed into the front end and displayed on the graphs. We compare the resulting graphs to the corresponding graphs on the RFSoC Studio Spectrum Analyzer, which comes with the board and serves as our ground truth.

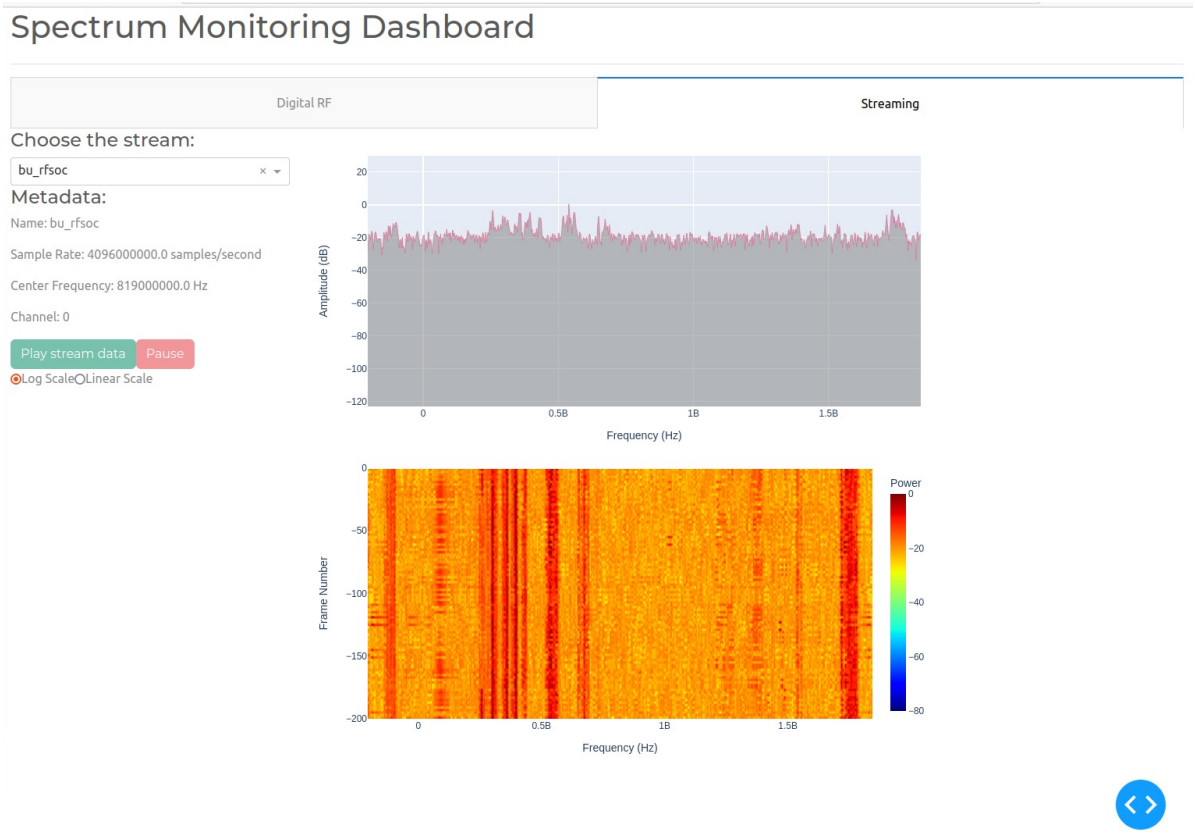

Figure 2: Spectrum Data Collected From Antenna, displayed on our web application

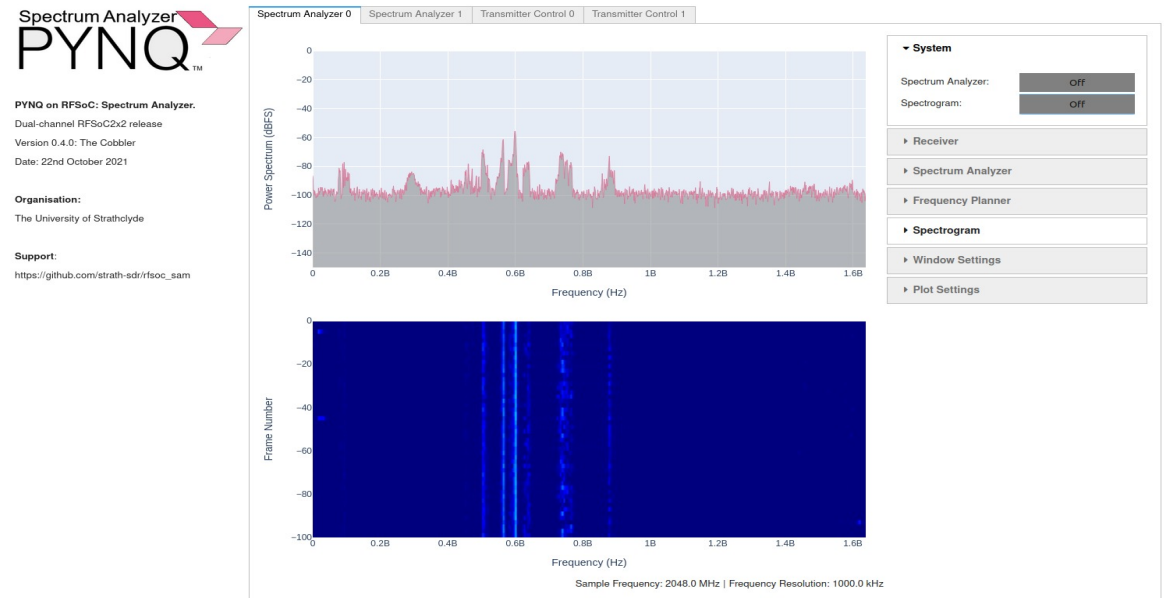

Figure 3: Spectrum Data Collected From Antenna, displayed on the RFSoC Spectrum Analyzer program

Though Figure 2 and Figure 3 have different y axes, it is evident that they are both showing similar sets of data, which means our web application is processing and displaying data correctly! However, it is important to mention that the data being displayed in Figures 2 and 3 were collected at slightly different times, so they are not showing the exact same sets of data. Both our application and RFSoC Studio have notable peaks around 600 MHz. However, there is an issue with the x-axis range, which affects all of our trials with Pipeline 1. The RFSoC Studio graph has an x-axis ranging from 0 to around 1.65 GHz, however ours starts off *below* 0 Hz. This issue needs to be investigated further.

<u>Pipeline 1, Trial 2: Loopback mode</u>

In this trial, the RX1 channel is connected to the TX0 output channel, and we control the frequency and amplitude of the signal transmitted by TX0. The resulting data is transmitted into RX1 and displayed in our web application.

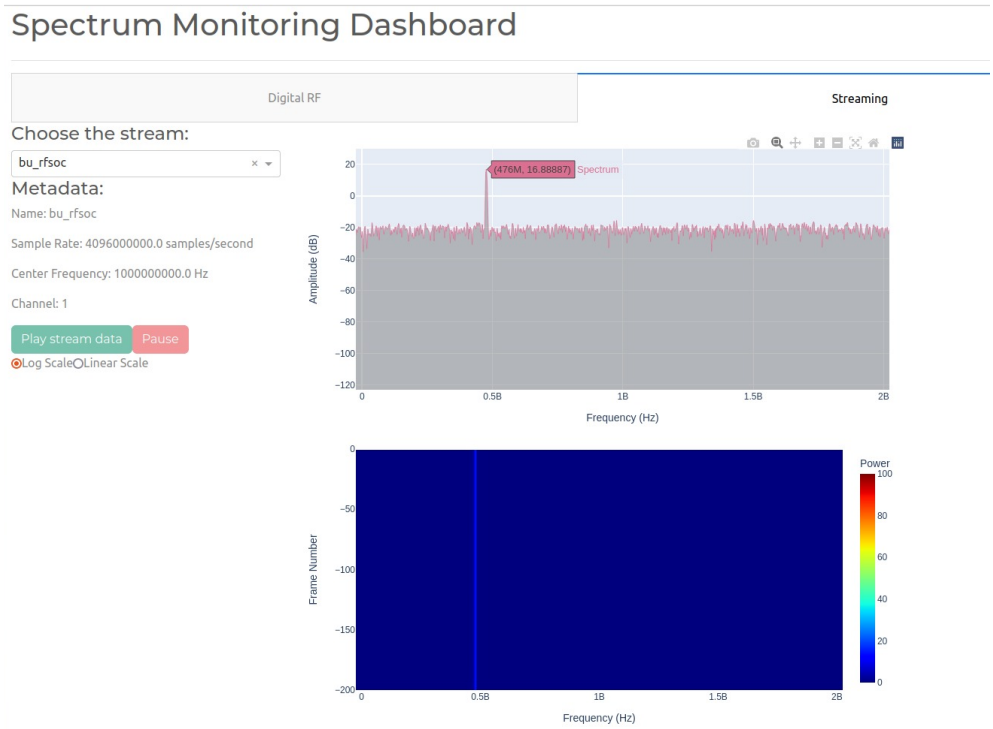

Figure 4: Spectrum Data Collected From Programmed Output Channel, displayed on the RFSoC Spectrum Analyzer program

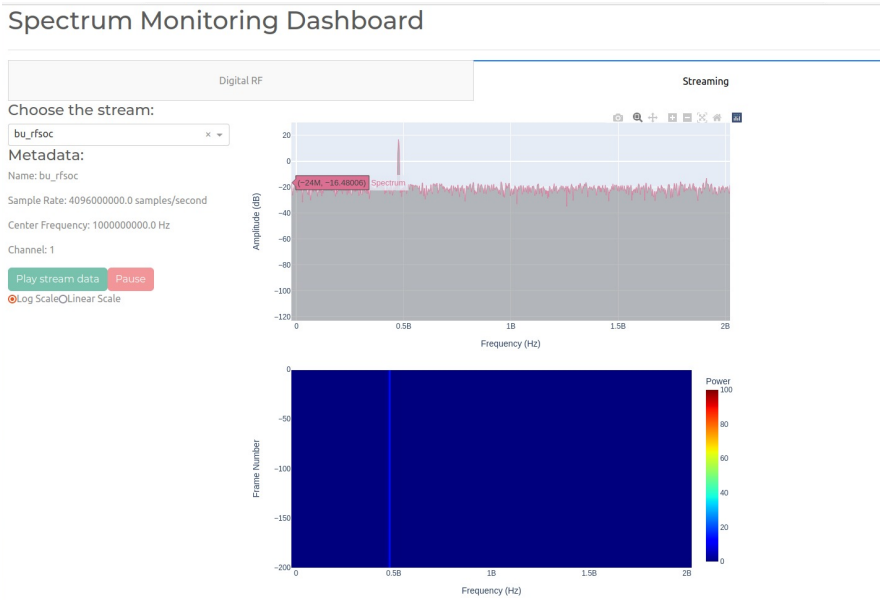

Figure 5: Spectrum Data Collected From Programmed Output Channel, displayed on the RFSoC Spectrum Analyzer program. Note that the minimum value of the x axis is -24MHz

In this iteration, we set the output of TX0 to transmit a peak with a frequency of 500 MHz and 0.8 gain. As shown in Figure 4, on our web application, this peak was displayed at 476 Mhz, so it is slightly displaced from the true value. Figure 5 shows that

the minimum value of the x axis is at -24 MHz, the same amount of displacement from the underlying correct value as the peak. As mentioned in the previous section, this x-axis issue needs to be investigated further.

RFSoC Channel	Frequency Domain Data Transmitted to Redis (T/F)	Data Displayed on Dashboard (T/F)	Data Matches RFSoC Studio Baseline (T/F)	Data can be played, paused. Log scale can be toggled (T/F)
RFSoC RX0	T	T	Mostly T (x axis issue)	T
RFSoC Channel RX1	T	T	Mostly T (x axis issue)	T

Pipeline 2 Digital RF Data:

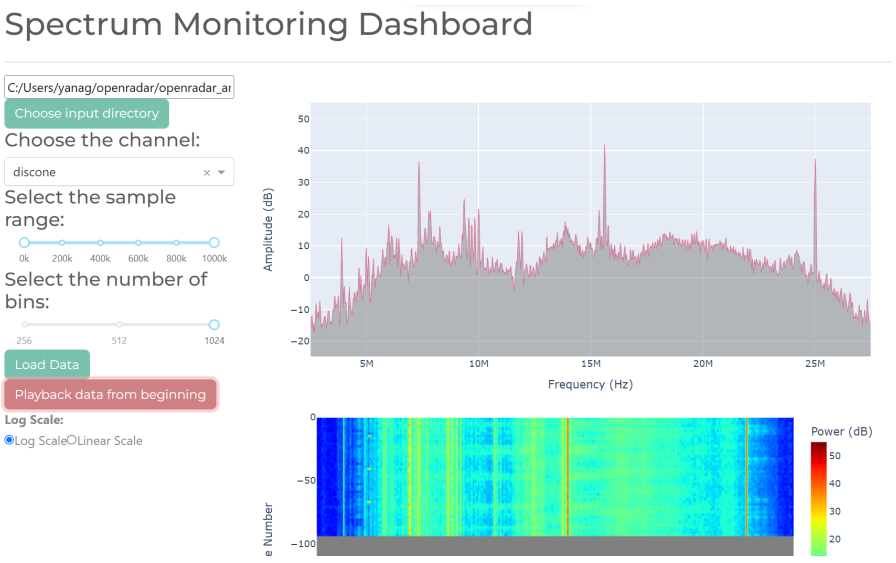

Figure 7: Display of Pre-recorded Digital RF Data

Figure 7 shows the spectrum plot of pre-recorded Digital RF data being processed through the web-app. The data shows various peaks which correspond to the different shades seen in the waterfall plot.

Digital RF Channel	The user can select the sample range, and number of bins (T/F)	Data played back properly (T/F)	Log scale can be toggled (T/F)
Discone	T	T	T
Monopole	T	T	T

Conclusions

We successfully tested both of our pipelines in this demo, however certain improvements need to be made before the final prototype testing.

Pipeline 1:
We were successfully able to stream live data from the RFSoC into our web application, however, some bugs, such as the aforementioned issue with the x-axis remain to be fixed. Next steps here include fixing these bugs, and adding more user interactivity.

Pipeline 2:
We were successfully able to display previously recorded DigitalRF data in our web application with no major bugs. All of the features, such as Next steps for this part of the project would be to improve the UI and to add more interactivity.