RFSoC for RF Environment Monitoring

Team 26 Critical Design Review
3/22/22

Yana Galina, Jaime Mohedano Aragon, Kakit Wong, Isaac Yamnitsky

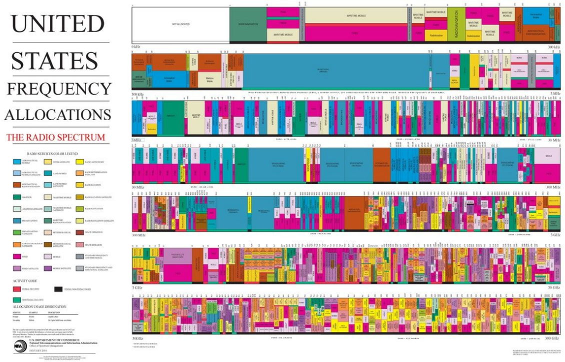
In collaboration with

MIT
HAYSTACK
OBSERVATORY

Problem Introduction

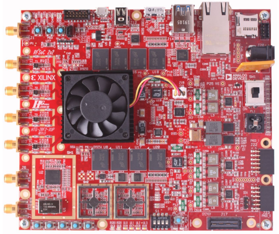
- The RF Spectrum is becoming increasingly congested
  - difficult for researchers in radio astronomy and geoscience to make measurements necessary for their work
- RF interference mitigation techniques are essential, and they require being able to monitor the wideband RF spectrum
- The Xilinx RFSoC board shows particular promise for monitoring due to the combination of its wide bandwidth, relative low cost, and ease of use

→ Solution: create an interactive web application with a variety of RF spectrum monitoring tools



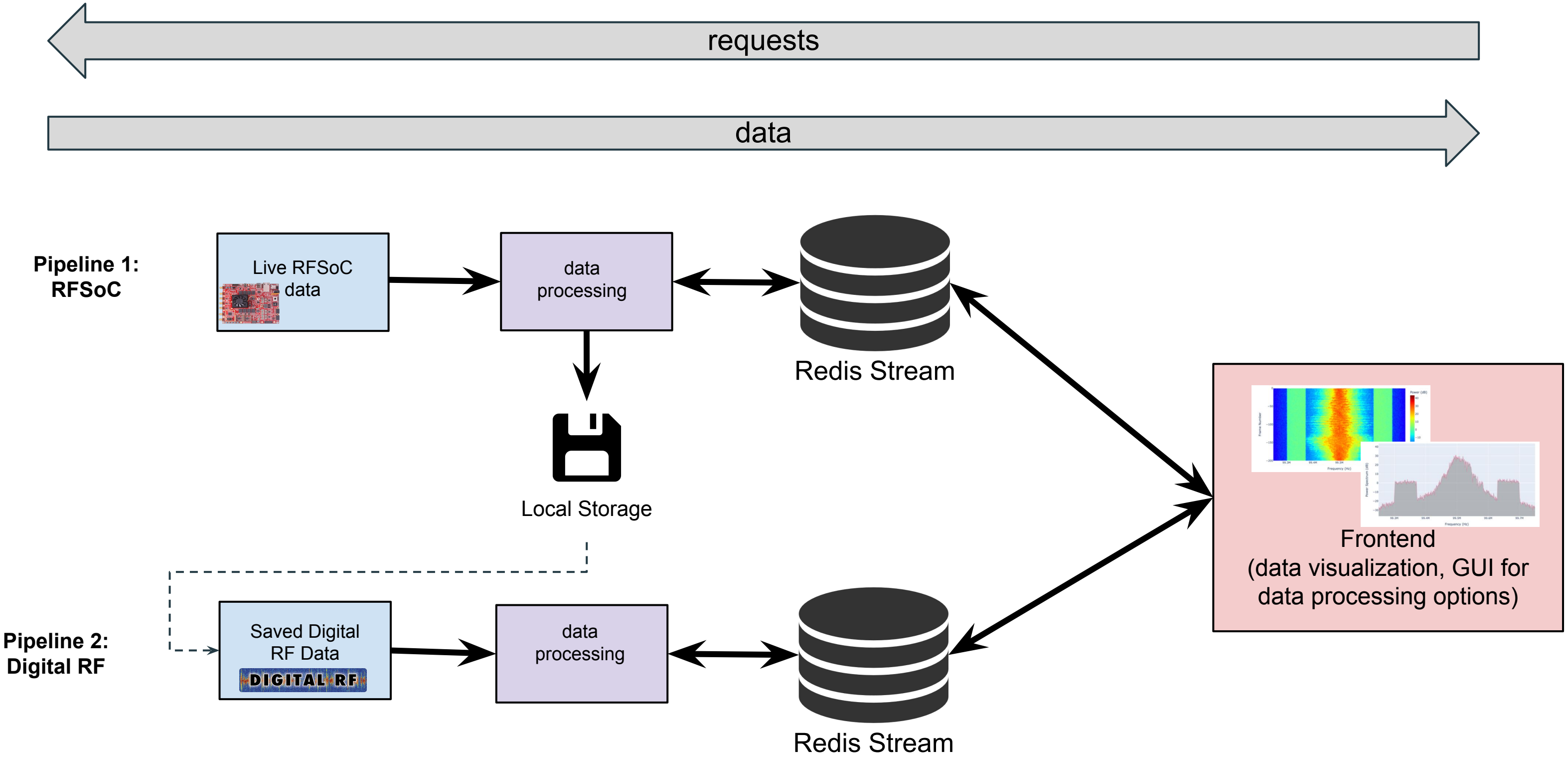
Overview

- End goal: aiding our clients’ RF interference mitigation research
  - We aim to create a base application that our clients can extend as needed
- Our project has split into two related but separate pipelines
  - Processing data in the Digital RF format
  - Live streaming data from the Xilinx RFSoC board
- Deliverables:
  - A web application using the Dash framework and Plotly graphs
  - Python scripts for the Xilinx RFSoC Board for pulling, processing, and storing data
  - An API for passing data and requests from the back end/board to the front end via a Redis Database



DIGITAL RF

System Block Diagram



requests

data

Pipeline 1: RFSoC

Live RFSoC data

data processing

Redis Stream

Local Storage

Pipeline 2: Digital RF

Saved Digital RF Data

data processing

Redis Stream

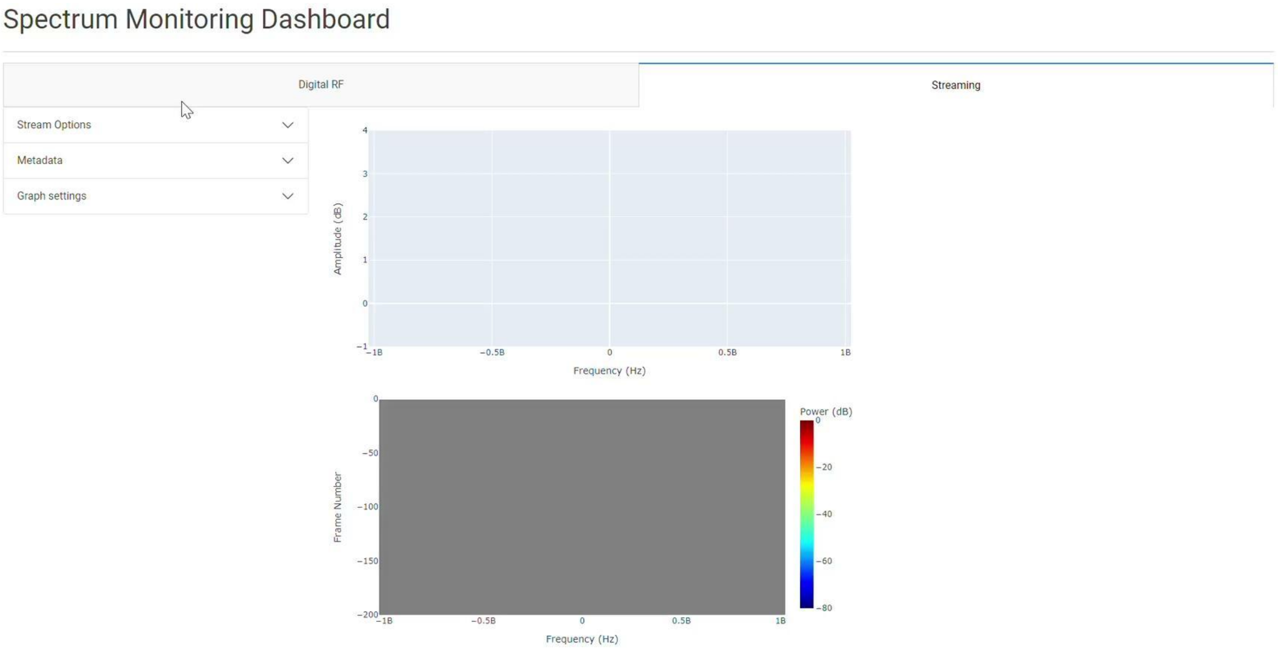
Frontend
(data visualization, GUI for data processing options)

Front End — Completed Work

- Spectrum and waterfall spectrogram plots
- User can toggle axis scales, track max/min points, change color scheme of graph
- Metadata is displayed
- Digital RF Playback
  - Able to play back stored digital RF data
  - User can select channel and other options
  - User can pause, play, and rewind data
- Live Streaming
  - User can select between any currently active stream
  - Able to stream live data
  - User can pause and play data

Demonstration – Digital RF playback

Demonstration – Simulated Live Streaming



Front End — Remaining Work

- UI improvements
- Improving robustness, error handling
- Adding in commands/requests for active interaction with RFSoC board
  - Current streaming is “passive” - only pulling data
- Adding a request form for downloading Digital RF data from the board

Redis Communication – Digital RF Playback

time

Front End	Back End
	Server initializes “request-id” to 0
	Server subscribes to all channels with the format “requests:*”
Client increments “request-id”, sends request for DigitalRF Channel Data using by pushing requested file to the channel “requests:<req_id>:channels”	
	Server responds with list of channels in “responses:<req_id>:channels”
Client obtains list of channels in “responses:<req_id>:channels”.	
User selects options for playback (number of bins, etc), client increments “request-id”, and sets request parameters in “requests:<req_id>:data”	
	Server sends metadata for request in the channel “responses:<req_id>:metadata”
Client obtains metadata for request in “responses:<req_id>:metadata”	
	Server dumps the DigitalRF data in the channel “responses:<req_id>:stream”
Client reads and displays each piece of data from “responses:<req_id>:stream”	

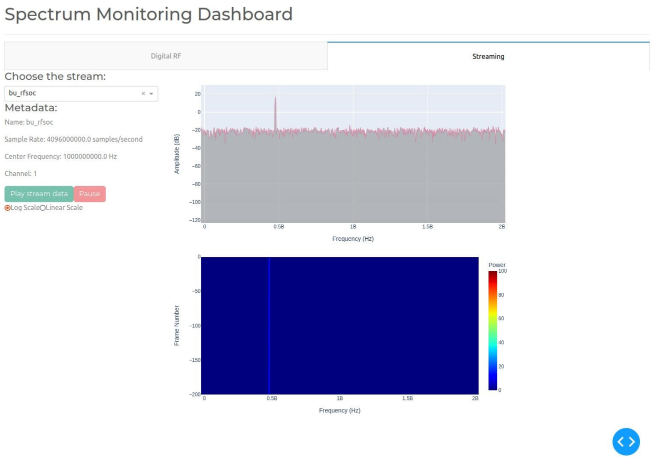
Redis Communication – Streaming

time

Front End	Back End/Board
	Back end adds name of stream to “active_streams” and dumps metadata in “metadata:<name>”
Client gets names of currently active streams from “active_streams”	Back end dumps data as it comes in, in a size-capped stream in the channel “stream:<name>”
User picks a stream, client gets metadata from “metadata:<name>”	
Until user hits “pause” or selects a new stream, the client repeatedly obtains the newest piece of data from the channel “stream:<name>” and displays it	

Board — Completed Work

- Successfully pulling raw data off board
- Converting raw data to spectrum data using Fast Fourier Transforms
- Pushing spectrum data and metadata to remote Redis Server
- Testing the board in loopback mode and with antennas connected



Spectrum Monitoring Dashboard

Choose the stream: bu_rfsoc

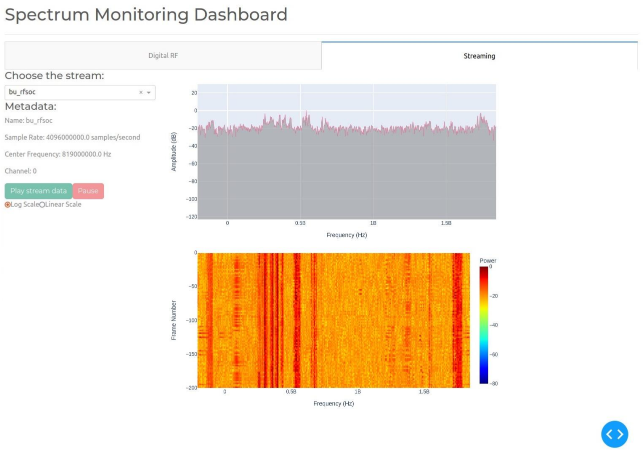
Metadata:

Name: bu_rfsoc

Sample Rate: 4096000000.0 samples/second

Center Frequency: 1000000000.0 Hz

Channel: 1



Spectrum Monitoring Dashboard

Choose the stream: bu_rfsoc

Metadata:

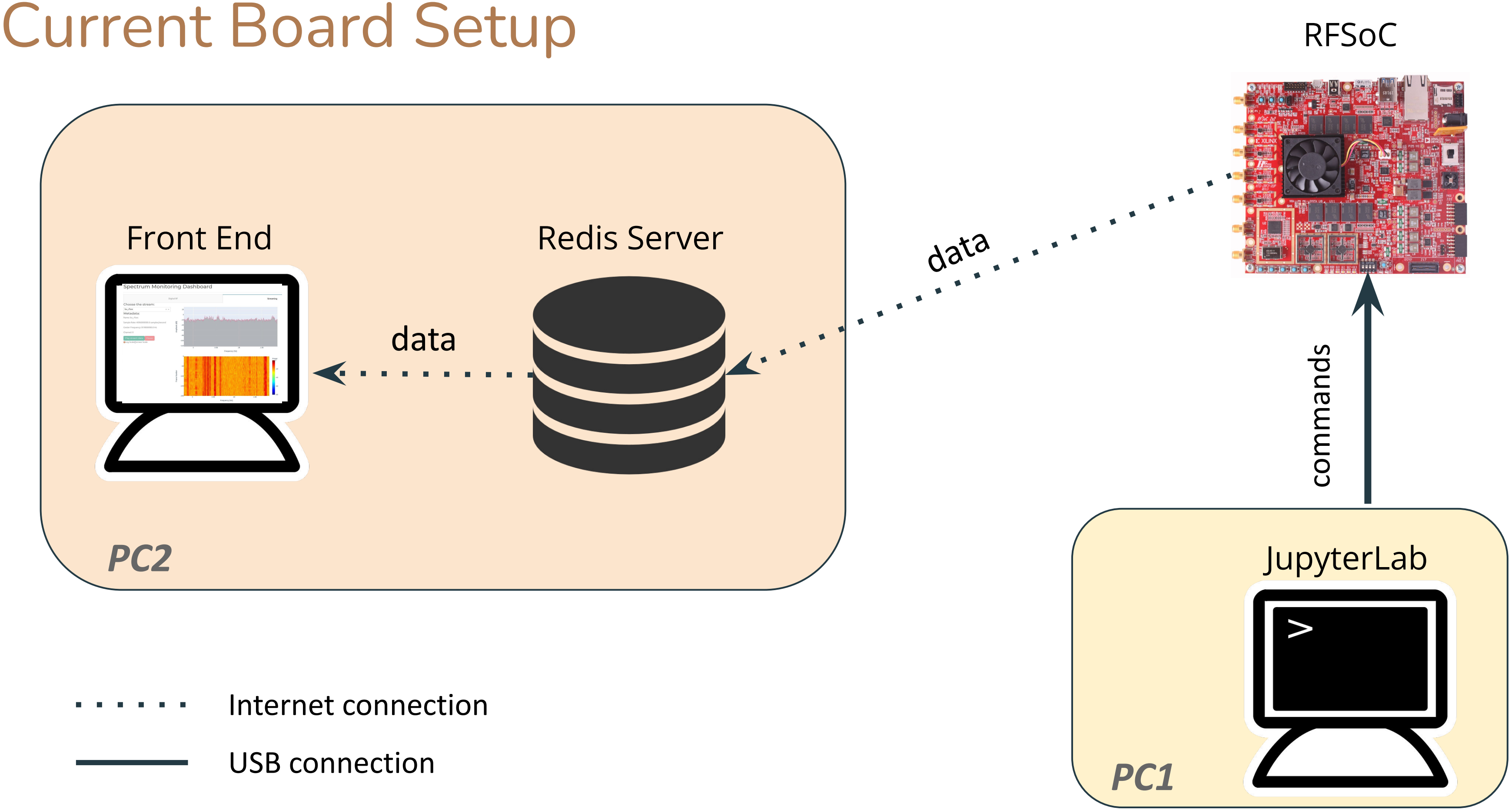
Name: bu_rfsoc

Sample Rate: 4096000000.0 samples/second

Center Frequency: 819000000.0 Hz

Channel: 0

Current Board Setup



RFSoC

Front End

Redis Server

data

data

commands

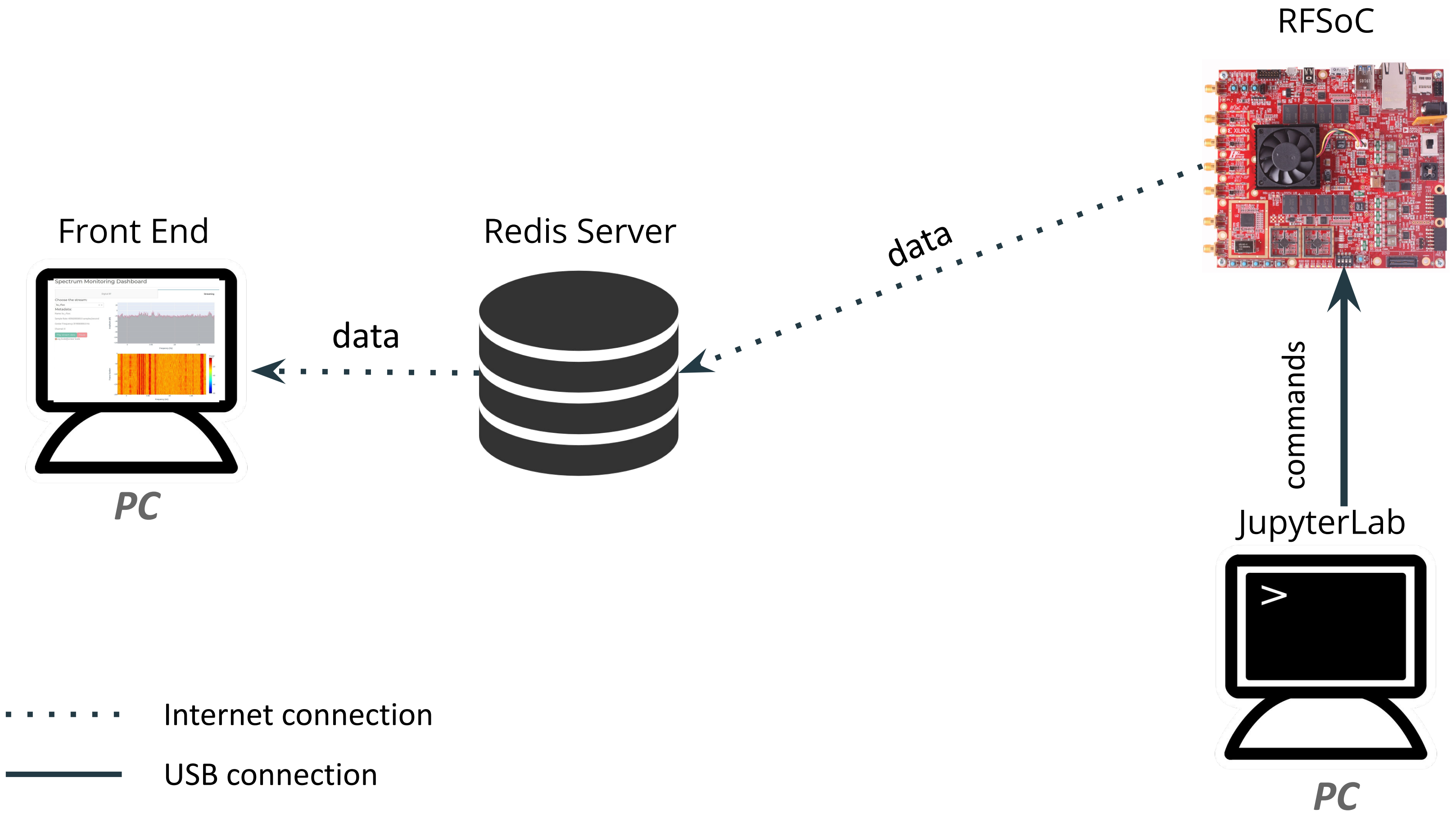
PC2

JupyterLab

PC1

Internet connection

USB connection



RFSoC

Front End

Redis Server

data

data

commands

JupyterLab

PC

PC

· · · · · · Internet connection

— USB connection

Board — Remaining Work

Primary

- Saving live RFSoC data in Digital RF format and being able to select which part of the spectrum to save
- Streaming a band and time limited set of raw IQ data

Secondary

- Accelerating the Numpy FFT function with the PYNQ framework by taking advantage of the hardware in the FPGA of the board
- Digital Down Conversion of the data
- Having the two receivers on, capturing each one a Nyquist Zone (0-2, 2-4) GHz

Gantt Chart

WBS NUMBER	TASK TITLE	START DATE	DUE DATE	PCT OF TASK COMPLETE	January					February				March					April					May		
					1	2	3	4	5	1	2	3	4	1	2	3	4	5	1	2	3	4	5	1	2	3
1	**Software**																									
1.1	Digital RF Playback through Redis Server	1/12/22	2/4/22	100%		■	■	■	■	■																
1.2	Simulating live stream in back end	2/5/22	2/10/22	100%						■	■															
1.3	Adding live stream capability to front end	2/11/18	3/1/22	95%							■	■	■	■												
1.4	Arranging UI, additional interactivity	3/2/18	4/7/22	90%									■	■	■	■	▨	▨	▨							
2	**Hardware**																									
2.1	Basic setup (connecting, running demo)	1/12/18	2/1/22	100%		■	■	■	■	■																
2.2	Pulling raw data off board	2/2/22	2/16/18	100%						■	■															
2.3	Converting raw data to spectrum data	2/17/22	3/3/22	100%							■	■	■	■												
2.4	Pushing spectrum data and metadata into Redis server	2/17/22	3/3/22	90%							■	■	■	■												
2.5	Converting board data to DigitalRF	3/4/22	4/7/22	25%										■	■	■	▨	▨	▨							
3	**Integration**																									
3.1	Displaying live data from RFSoC on front end	2/17/22	3/3/22	90%							■	■	■	■												
3.2	Second Prototype Testing	3/3/22	3/3/22	100%										■												
3.3	Requesting and downloading Digital RF Data from RFSoC	3/4/22	4/7/22	0%										■	■	■	▨	▨	▨							
4	**Testing, Installing, Presenting**																									
4.1	Final Prototype Testing	3/29/22	4/7/22	0%															▨							
4.2	Customer Installation	4/7/22	5/5/22	0%															▨	▨	▨	▨	▨	▨		
4.3	ECE Day	5/6/22	5/6/22	0%																				▨		

Thank you!

Questions?