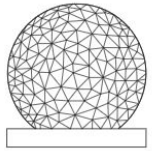
Team 26 PDR

RFSoC for RF Environment Monitoring

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

12/2/21

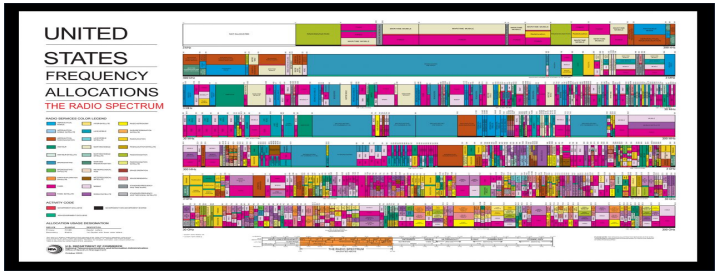
In collaboration with



MIT HAYSTACK OBSERVATORY

Introduction

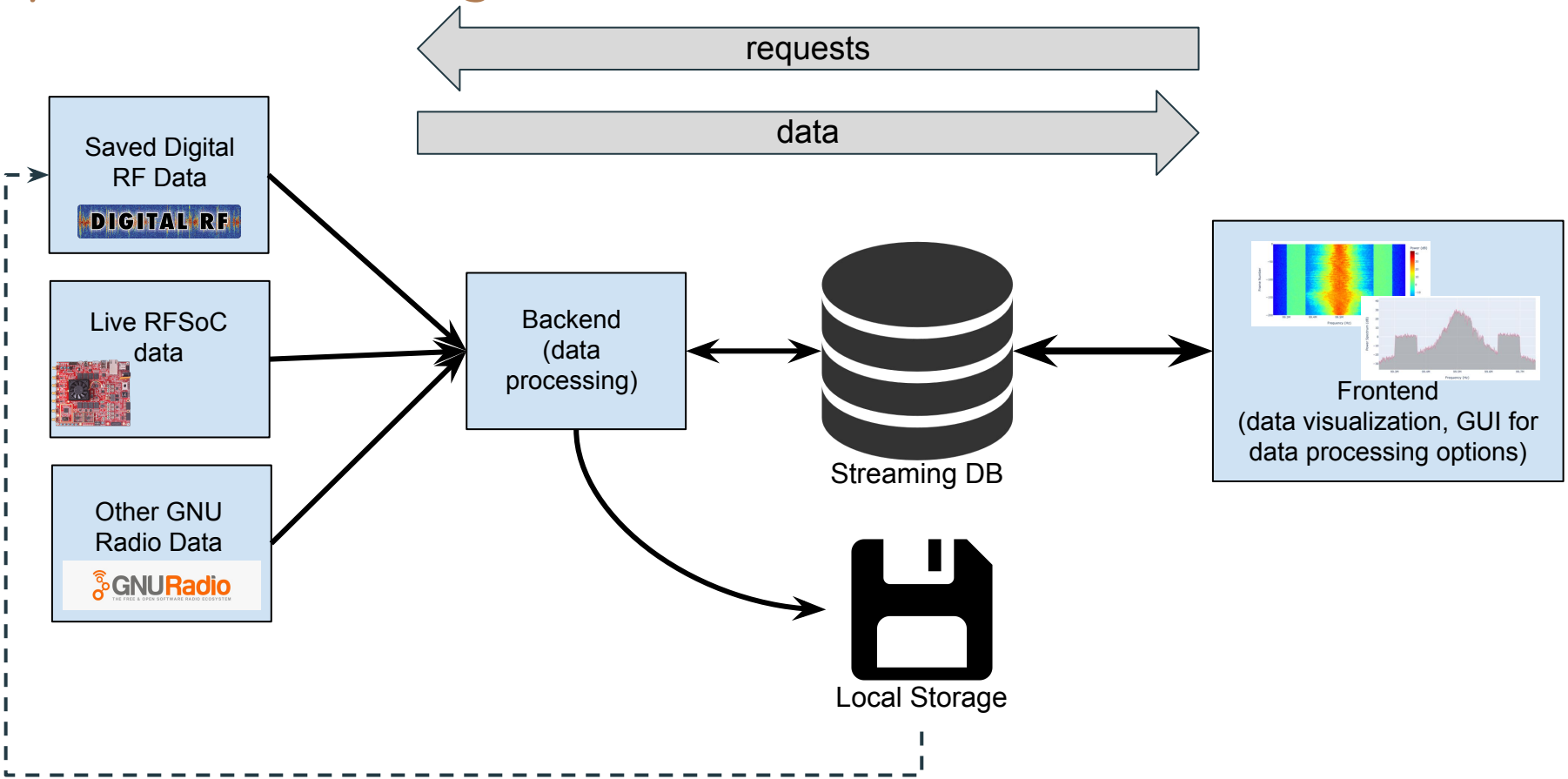
- The RF Spectrum is becoming increasingly congested
  - difficult for researchers in radio astronomy and geoscience to make measurements necessary for their work
- RF interference mitigation techniques are essential
  - require being able to monitor the wideband RF spectrum
- → Solution: create an interactive web application with a variety of RF spectrum monitoring tools
  - end goal of aiding our clients' RF interference mitigation research

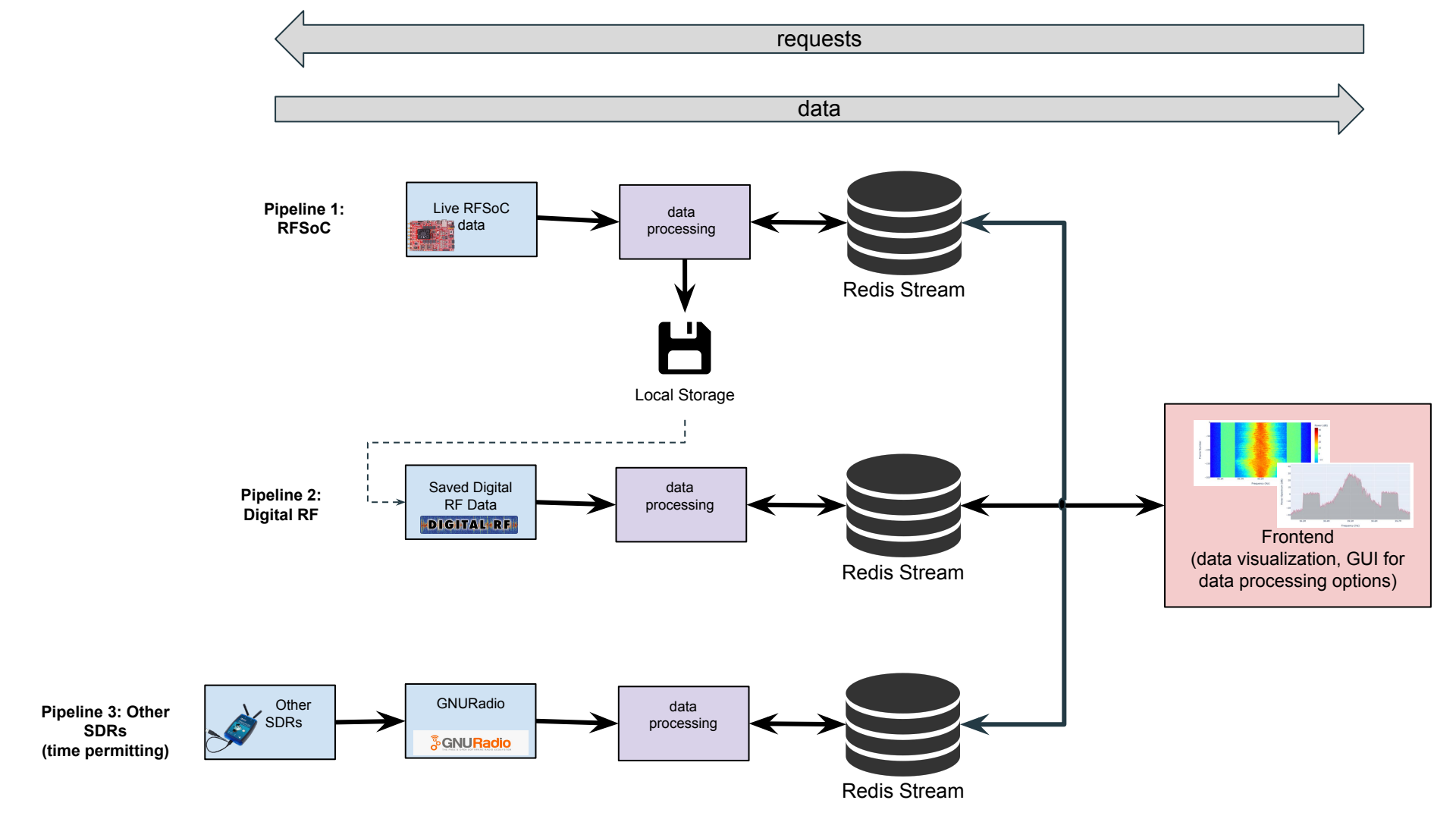


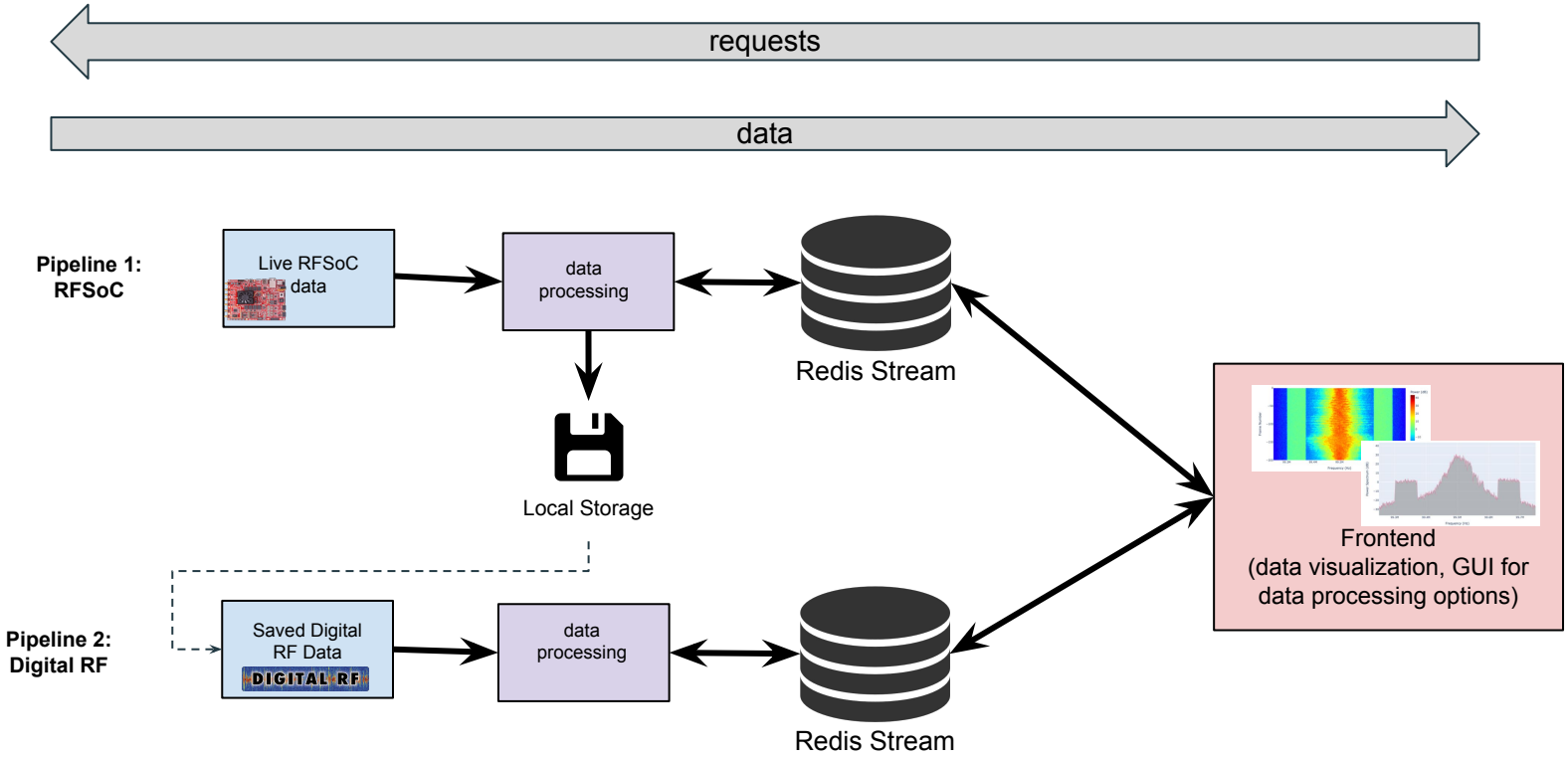
Requirements

- A web application which will use a variety of RF data sources to display the RF spectrum, efficiently parse and store data, and support demodulation for certain RF signal types
  - Utilizing Xilinx RFSoC 2x2
- Should be able to display and process:
  - Live RFSoC data
  - Stored DigitalRF data
  - Other live data through GNURadio (time permitting)

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
DIGITAL RF
data
processing
Redis Stream
Frontend
(data visualization, GUI for
data processing options)
Pipeline 3: Other
SDRs
(time permitting)
Other
SDRs
GNURadio
GNURadio
data
processing
Redis Stream


requests
data
Pipeline 1:
RFSoC
Live RFSoC
data
data
processing
Redis Stream
Local Storage
Frontend
(data visualization, GUI for
data processing options)
Pipeline 2:
Digital RF
Saved Digital
RF Data
DIGITAL RF
data
processing
Redis Stream

Front end


Spectrum Monitoring Dashboard
C:\Users\yanag\openradar\openradar_antennas_nb_vhf
Choose input directory
Choose the channel:
discone
Select the sample range:
0k
200k
400k
600k
800k
1000k
Select the number of bins:
256
512
1024
Metadata:
Sample Rate: 500000.0 samples/second
Center Frequency: 99499999.99415477 Hz
Channel: discone
Load Data
Playback data from beginning
Log Scale:
Log Scale
Linear Scale
Power Spectrum (dB)
Frequency (Hz)
Frame Number
Power (dB)

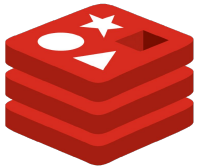
Front End:

- The most developed part of our project so far
  - Displays DigitalRF data with spectrum and spectrogram plots
- Responsible for visualizing data and displaying options for user
- Sends requests to backend for data
- Parses incoming messages containing data for visualizations
- Language: Python
  - Easy to work with, many relevant tools and libraries exist for development
- Graphs: Plotly
  - Easy to make interactive graphs, similar (open source) spectrum graphs using Plotly found in StrathSDR which comes with RFSoC board
- Dashboard: Dash Framework
  - Integrates well with Plotly graphs, intended specifically for complex dashboard visualizations

Backend:

- Responds to requests from frontend with corresponding data
- Responsible for data collection and (part of) processing
  - Reading raw DigitalRF files and converting them into power readings that can be used for spectrum graphs
  - Converting RFSoC data to DigitalRF Data
  - Much data processing for RFSoC will be done on-board for hardware acceleration benefits
- Language - (mainly) Python
  - RFSoC board has tools specifically for integration with Python (PYNQ Framework)
  - Development is fast
  - Modules in C/C++ can be added if needed for efficiency

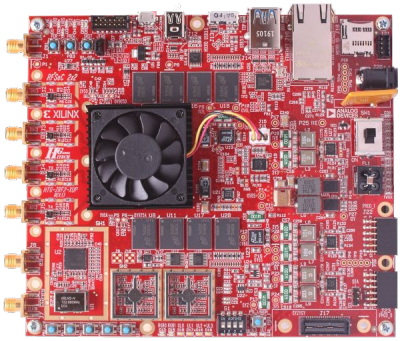
Redis Streaming Database



redis

- Redis Streams - an in-memory data structure which allows for streaming operations between producers and consumers
- Streams data and commands between front and backend
  - Provides a single clean interface for the frontend
- Well suited for time series and message queues
  - great for updating graph data over time
- Fast - Redis latency tests show 99.9% of requests have a latency <= 2 milliseconds
- Have not incorporated this yet - still designing message structure

Main Hardware - Xilinx RFSoC

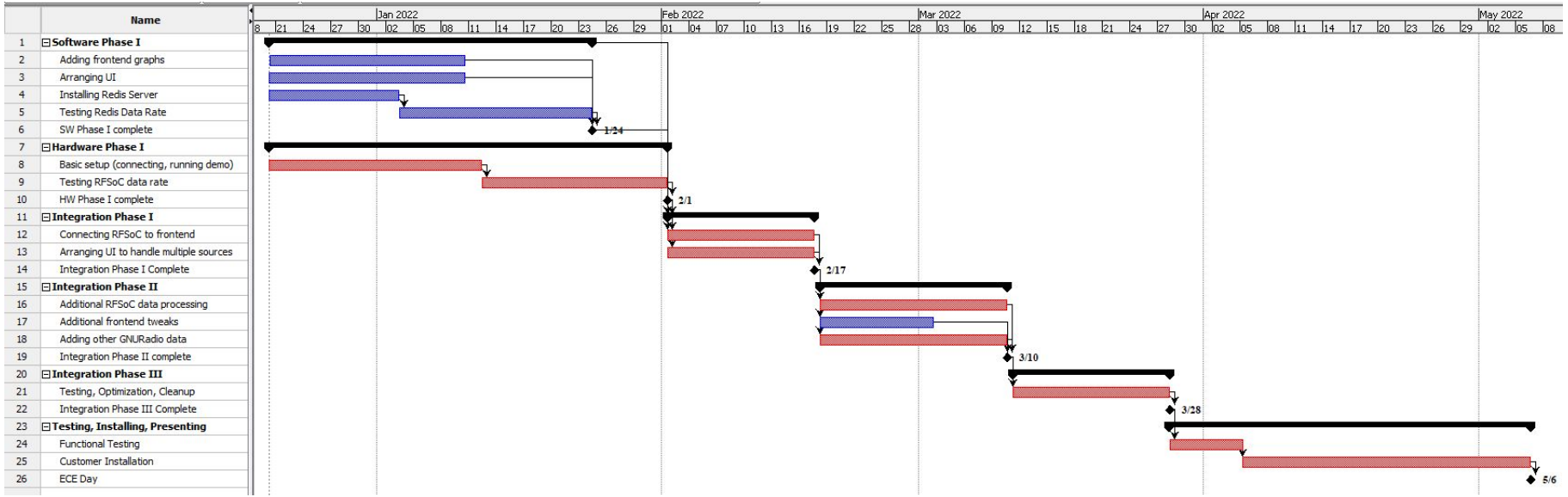


- Major roadblock - have not received access to one yet!
  - Should be able to access client's board remotely as of this week
  - Arrival date of group's board is still unknown
- Powerful board with large bandwidth and data processing capabilities
- Will be used to collect, process, and store raw signal data in real time
- Will benchmark the board to test the rate of raw signal data that can be collected and stored
- Will utilize the PYNQ framework for hardware-accelerated data processing, some additional processing may be done by backend

Other Data Sources

- DigitalRF data
  - Previously recorded, locally stored data which is fed into the backend to be processed and displayed
- GNURadio
  - An open source framework built for SDR applications
  - Will be used together with a board (ADALM-PLUTO) as a live data source to be fed into the backend running on a host computer

Schedule


Name
Jan 2022
Feb 2022
Mar 2022
Apr 2022
May 2022
1 Software Phase I
2 Adding frontend graphs
3 Arranging UI
4 Installing Redis Server
5 Testing Redis Data Rate
6 SW Phase I complete
7 Hardware Phase I
8 Basic setup (connecting, running demo)
9 Testing RFSoC data rate
10 HW Phase I complete
11 Integration Phase I
12 Connecting RFSoC to frontend
13 Arranging UI to handle multiple sources
14 Integration Phase I Complete
15 Integration Phase II
16 Additional RFSoC data processing
17 Additional frontend tweaks
18 Adding other GNURadio data
19 Integration Phase II complete
20 Integration Phase III
21 Testing, Optimization, Cleanup
22 Integration Phase III Complete
23 Testing, Installing, Presenting
24 Functional Testing
25 Customer Installation
26 ECE Day
1/24
2/1
2/17
3/10
3/28
5/6