CartoFreq: Visualizing RF Signal Strength in Real-time

Cody Roberts
Chicago, IL
codyjroberts@gmail.com

Mac Carter
Chicago, IL
harlow44@hotmail.com

Brian Hamilton
Chicago, IL
bhamilton292@gmail.com

Abstract
This document acts as a detailed insight into the functionality and utilization of the CartoFreq visualization project for displaying and recording live-data streams from Keysight's N6841A RF Sensor.

Introduction

CartoFreq is a tool for quickly assessing signal strength from a range of frequencies at a particular location. Designed to work with Keysight's N6841A RF Sensor, CartoFreq monitors a range of frequencies gathered by the device, naively converts them to a relative signal strength, and displays them in a sunburst-esque manner on a map. The N6841A is equipped with GPS allowing CartoFreq to track the sensors location in conjunction with the frequency data. With this in mind we designed CartoFreq to be a mobile tool, allowing the user to view the data on their mobile device while carrying the sensor on one's person. We also included the additional feature of recording and playing back data, giving the user the ability to analyze the signal strength in further detail whether in the field or the office.

Related Works

Our visualization techniques drew inspiration from the traditional wifi signal indicator, converting it into a sunburst chart to handle multiple signals. The stacked streamgraph is fairly standard with the weight shifted to the bottom.

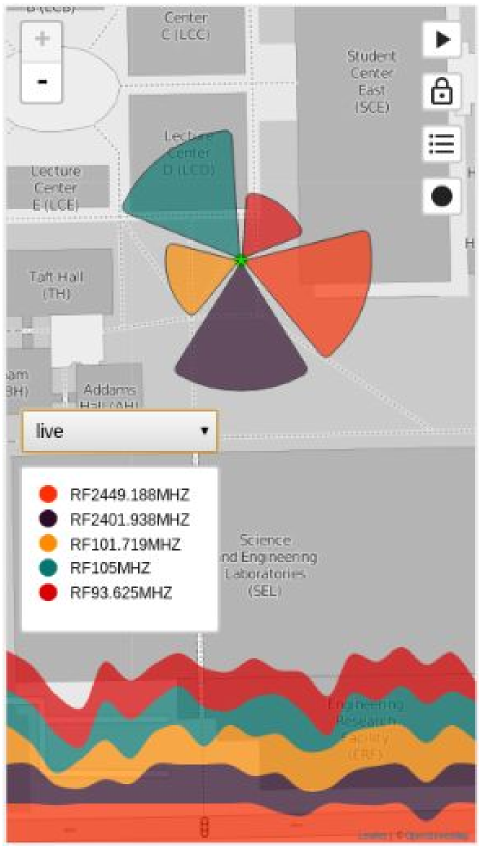


Fig 1. A demonstration of live data

A proprietary library, AgSlLib, contained all of the functionality for polling the sensor that we used, along with a few basic examples of GPS, FFT and sensor polling.

Getting live data to our client-side application was done with the usage of the Flask and SocketIo frameworks for python.

Analysis Tasks

Our task was to create visualizations that show the intensity of particular RF data at different locations in real-time for analysis. Not only does CartoFreq present the real-time RF data, but it also displays the recent previous data via markers on a single visualization.

CartoFreq visualizes the presence and magnitude of frequencies pertaining to 5ghz wifi, 2.4ghz wifi, fm radio, am radio, tv1 (channels 2-6), and tv2 (channels 7) at the physical particular location of the RF sensor in real-time. The data is visualized via a sunburst pie chart on a map centered at the sensor location (derived from the latitude and longitude coordinates sent with the data), where the magnitude of each frequency is denoted by the size of the corresponding slice of the sunburst pie chart. As the data is streamed in, the sunburst pie chart updates with each new piece of data, while the previous data is presented as a streamgraph at the bottom of the screen. This streamgraph dynamically updates as each new chunk of data arrives to show the 20 most recent values and provide some historical context of the stream. Historical markers are also placed on the map at the previous 30 locations of the RF sensor equipped with all of the frequency magnitude information.

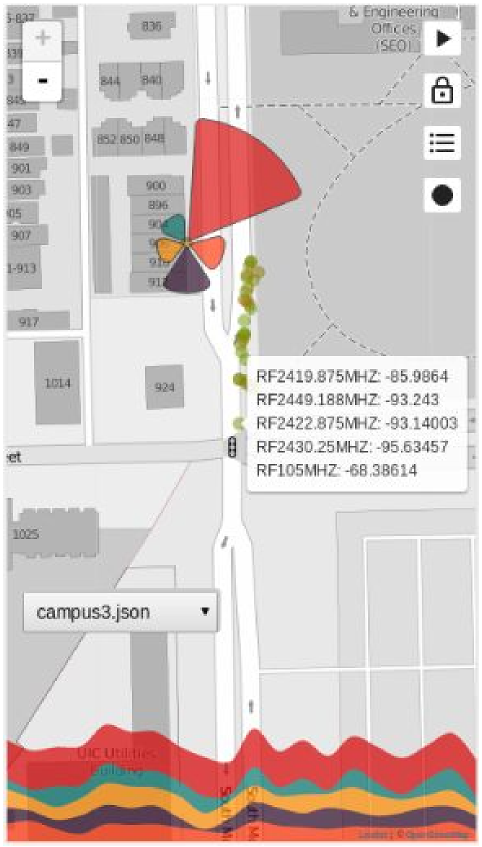
With the record, pause/play buttons, and selection box for displaying either live or recorded data, CartoFreq is a tool to clearly display, record, and playback data with ease.

Visualization Details

Polling data was done with C# and a proprietary library built for our sensor, AgSilLib. Then using the data for multiple bands on a particular frequency we built up a custom averaging of the signal, also known as an FFT. This was shown in some example code but code was rewritten for this project for readability and reliability. The data was saved off to files, for sending to a connected client.

Python was used as our server, we utilized the SocketIo and Flask libraries. Code was written for compatibility with both 2.7 and 3.5. Files are read from the directory where the sensor is working and broadcast out to connected clients over port 8088. We used JSON to send the data to our client. Our code is written to update our client every 1/4th of a second. The server is also capable of recording live data for later retrieval. In addition to the server we wrote a fake sensor in python so we can generate data to show it in action without always having a sensor around to demo it. This is for demo purposes only, as we might not have the ability to connect inside UIC's network unless the correct ports are open.

Clients connect in a Javascript webpage designed primarily for Mobile devices. We used D3 v4, SocketIo and Leaflet to help us display our data. Leaflet gives us a nice map interface to overlay our D3 graphs , controls and labels. SocketIo talked to our Python server very nicely and we parse our JSON return data into our visualization. A timeline is kept at the bottom so users can see the last 30 seconds of data. The client also has the ability to get the exact sensor readout for a particular category frequency by hovering over it. The user can also choose to make recordings of the data that they are seeing up to 100 iterations, which logs to the server, to be played back on any connected client.



RF2419.875MHZ: -85.9864
RF2449.188MHZ: -93.243
RF2422.875MHZ: -93.14003
RF2430.25MHZ: -95.63457
RF105MHZ: -68.38614

Fig 2. Historical markers encode overall sensor signal strength as well as provide detailed information on hover.

Case Study

To utilize the majority of the features of CartoFreq, one can begin by setting up the livestream between the RF sensor and the client. Setup can be configured using either the live data or the spoofed data, detailed in the README.md file included in the CartoFreq root project directory.

After this setup is complete, one may select the "live" option from the drop-down selection list located on the left side of user interface. Once "live" is selected, the user will see the live data update on the screen as the frequency magnitude values change between each time interval. At this point, one may select the black dot "record" button at the top-right of the screen, at which point the icon turns red and the data is written to a file on the remote server.

Once one feels that a satisfactory amount of data has been recorded, they can click on the record button a second time to end the recording process. At this point, an option will be added to the same drop-down selection list that is currently set to "live" with the filename of the newly written file (titled after the time-stamp corresponding to the time at which the file was written.

Now the user can select this newly written from the drop-down selection list and it will be loaded onto the user interface and begin visualizing the data on the display. At any point during playback, one can press the pause button at the top right of the screen. When paused, the recording temporarily stops playback and freezes on the current iteration through the recorded data. Hovering over the markers on the map results in a tooltip that indicates the magnitude values for each displayed frequency at that particular time when the mark was first added. The recording can resume playback at the current point by selecting the play button which has replaced the pause button at the same location in the top right of the screen.

This recording and playback functionality can be implemented at any point while viewing live data. There is no limit to the number of files you can record and add for easy playback.

Expert Feedback

Unfortunately we were unable to speak with any domain specialists during the development of CartoFreq which has led us to making design decisions based on guess work. We speculate that the application could be used to locate ideal positions for placing transponders and receivers, or to discover possible sources of interference. We are open to continuing our work if interest is shown and in that spirit we developed the application for modularity and flexibility, giving us the ability to add and remove features with little effect on the rest of the program.

Conclusions and Future Work

While we were not able to deliver on each and every of our many ideas, we were able to produce a visualization tool which implements the core functions we sought to create. The main component of our original visualization idea was to stream live data from the RF sensor to our client. This streaming implementation required a significant effort that we were not able to accomplish until several days before the deliverable was due. Thus, we were unable to fully realize some of our other ideas, such as an axis contains time stamps of the data for both live and recorded data. Regardless, we truly believe that CartoFreq is a useful tool for its live updating of data, its ability to record and reproduce data streams, and its simple, easy to interpret data visualizations to show changes in the magnitude of frequencies both at and over particular time intervals.

CartoFreq is open source and work on it can be tracked on Github[1].

References

1. GSM Frequency Band. (n.d.). In Wikipedia. Retrieved Dec 5, 2016, from https://en.wikipedia.org/wiki/GSM_frequency_bands
2. WLAN Channels. (n.d.). In Wikipedia. Retrieved Dec 5, 2016, from https://en.wikipedia.org/wiki/List_of_WLAN_channels
3. Broadcast Band (n.d.). In Wikipedia. Retrieved Dec 5, 2016, from https://en.wikipedia.org/wiki/Broadcast_band

[1] https://github.com/codyjroberts/cartofreq