

CSE 523: Advanced Project
Advisor: Professor Samir Das
Student: Salman Qavi
Student ID: 109470856
Demo Date: Dec. 12, 2019

Project Report

The key idea behind the SpecSense project is to monitor spectrum using cheap spectrum sensors such as RTL-SDRs. Our team at WINGS Lab has published a number of research papers successfully demonstrating how to perform tasks such as overall spectrum monitoring, illegal transmitter detection and localization as well as sensor selection [1, 2]. All these tasks were evaluated by building an actual testbed using SDR-based transmitters as well as RTL-SDRs. However, running this testbed remained a cumbersome job, as the system requires a large number of inputs, such as IP addresses of the transmitters and the sensors, scripts to run the sensors for different locations of the transmitter. The objective of this project is to make it much easier to run the testbed with minimal manual input from the user.

To reduce such manual input, we design a web-interface based GUI as a frontend of our testbed. This web interface controls a distributed set of RTL-SDR sensors remotely over local area network. Our backend system can capture, store and fetch radio-frequency samples and use them for further analysis including plotting time-series graph against Power Spectrum Density values. The sensors are connected to Odroid-C2 devices [3] and the transmitters - either HackRF [4] or USRP [5] - are powered by laptops. We tested the web-interface indoors with up to eight Odroid C2 sensors and both USRP and HackRF transmitters.

The web-interface allows a user to discover sensors/transmitters and capture/fetch radio-frequency data by clicking a few buttons. As a result, all inconveniences of typing commands in terminal and passing arguments to functions manually through the command line are eliminated.

The web-interface accomplishes the task of radio-frequency data collection and visualization in five steps as described below:

1. discovers all RTL-SDR sensors connected to Odroid C2 devices in the same subnet from *collect_rx_data.html* (Figure 1)
2. discovers all transmitters powered by laptops from *index.html* (Figure 2)
3. requests all sensors via SSH to start sensing RF data from *allrx.html* (Figure 3)
4. requests transmitter to send RF signals of frequency and gain specified by the user at *tx_run.html* (Figure 4)
5. captures RF signals for the number of samples requested by the user, stores them locally on the Odroid C2, fetches Power Spectrum Density values for the samples, and plots a time-series graph at *chart.html* (Figure 5)

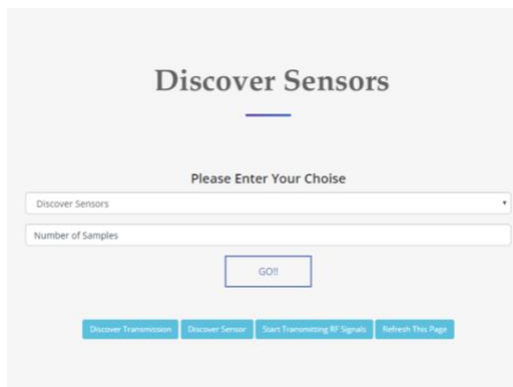


Figure 1: on clicking 'Go' at *collect_rx_data.html*, all sensors in the subnet are discovered and their hostnames and IP addresses stored

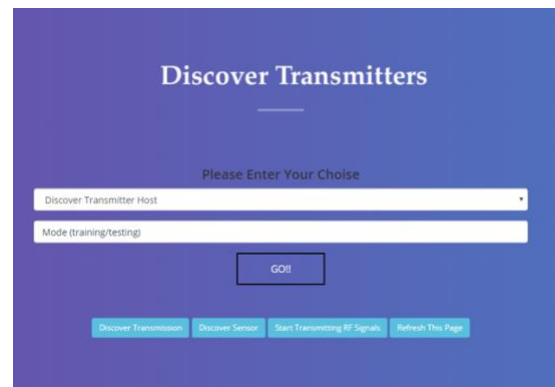


Figure 2: on clicking 'Go' at *index.html*, all transmitters – USRP or HackRF - in the subnet are discovered and their hostnames and IP addresses stored

Figure 3: at [allrx.html](#), users can specify the number of iterations, the duration between the iterations to sense RF data for. Additionally, they can request the Odroid C2 hosts to be powered off after sensing.

Figure 4: at [tx_run.html](#), users can choose between HackRF or USRP transmitters, the gain and frequency of the transmission, and the mode (i.e. 'test' or 'train') for the transmission of RF signals.

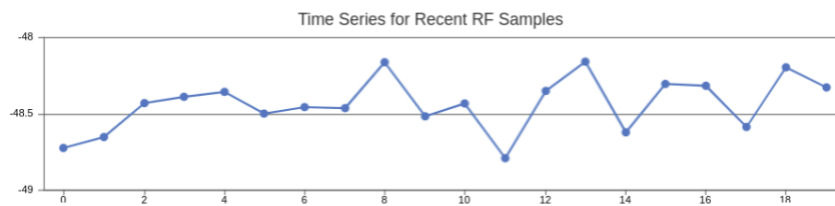


Figure 5: The user can fetch the RF data samples stored locally on the Odroid C2 devices and get a time-series graph against Power Spectrum Density values for the requested number of samples at [chart.html](#).

All server-side processing is handled by Python Flask wrapper, which is a common framework to design web applications. We used Ncrack, a high-speed network authentication cracking tool, for discovering all sensors and transmitters in the subnet and PDSH to communicate with multiple Odroid C2 hosts at once and retrieve output from them. Upon successful discovery, their hostnames and IP addresses are displayed on separate tables on the web-interface and permanently stored for future use. The interface supports both USRP and HackRF transmitters and users can select the frequency, gain, and mode (i.e. test or train) for the transmission of RF signals.

The backend is primarily built using Python Flask in addition to Ajax and jQuery for plotting graphs. The frontend uses HTML/CSS and JavaScript. The scripts are written in Python 3.6 and tested on Ubuntu 18.04. Running *mainDriver.py* on the terminal starts the local server, and the web-interface can be accessed by navigating to [localhost:5000](#) on any browser. Additionally, there is a homepage that describes the project work and how-to setup the testbed for its use both indoor and outdoor.

The web-interface will facilitate indoor and outdoor testbed experimentation for the study of localizing multiple intruders using a distributed set of sensors. In addition, the setup and web-interface can be used for the collection of big data using distributed set of sensors for RF analysis across multiple spectrum bands.

Demo Link:

https://www.youtube.com/playlist?list=PLCgCXLlrfGZt7ok3_UoM3DTh6nZk6-kp7

References:

- [1] A. Bhattacharya, C. Zhan, H. Gupta, S. R. Das, and P. M. Djuric, "Selection of Sensors for Efficient Transmitter Localization," *INFOCOM*.
- [2] A. Chakraborty, A. Bhattacharya, S. Kamal, S. R. Das, H. Gupta, and P. M. Djuric, "Spectrum patrolling with crowdsourced spectrum sensors," in *IEEE INFOCOM*, 2018.
- [3] "C2," *odroid*. [Online]. Available: <https://wiki.odroid.com/odroid-c2/odroid-c2>. [Accessed: 12-Dec-2019].
- [4] "HackRF One," *Great Scott Gadgets*. [Online]. Available: <https://greatscottgadgets.com/hackrf/one/>. [Accessed: 05-Dec-2019].
- [5] *USRP B200/B210 Information*. [Online]. Available: https://files.ettus.com/b2x0_enclosure/. [Accessed: 01-Dec-2019].