Project flooRFinder Design and Architecture

Using SDR to find noisy and quiet RF environments

Overview

- The HF band noise is so high I cannot participate in nets from home!
- How far do I need to travel to find a quiet RF environment?
- I bet I-5 is more RF noisy than I-405.

- These questions and more can be answered a number of ways.
- What if the noise and signal levels could be recorded and shared?

- This project aims to provide a solution.

Goals

- Create an SDR-based solution.
- Solution should (be):
  - Multi-platform (Windows and Linux)
  - Easily reproducible (documentation for others to follow),
  - Utilize SDR of any cost (though higher cost could lead to better results),
  - Locally store captured data,
  - Enable APRS reporting but not necessarily require it,
  - Limit costs in terms of involved components,
  - Open-source,
  - Easily portable,
  - Provide advice on antennas including type, placement, and limitations.

Hardware Overview (a work in progress)

- Software Defined Radio:
  - Any SDR radio could provide the hardware connection to the antenna.
  - RTL's SDRPlay used as design and build baseline system.
- Windows or Linux/Raspbian-based laptop or micro PC:
  - Keyboard, Monitor, USB ports, etc.
- Antenna System:
  - Baseline antenna was a 54" whip tuned for 50MHz band,
  - Feedline was LMR-240 to NMO lip-mount.

Software Overview

- SDR Uno:
  - Provides SDR hardware access
  - Displays and records Signal Strength and SNR
- Python:
  - Custom action scripts gather GPS data, SDRUno output, etc
- Triggers and Timing:
  - DireWolf provides automated packet reporting services
  - AT or CRON can be used to gather data locally
- APRS:
  - Direwolf. Other APRS solutions could work but were not in initial design.

System Assembly and Setup

- Create a Project Folder on you Desktop like: flooRFinder
- Python [file/s]: Put in a sub-folder like flooRFinder\Scripts
- SDRuno:
  - Software UI has a widget to find the hardware USB interface
  - Click/Enable ‘Power & SNR to CSV’
  - MAIN SP > Settings > CSV Filename > …\flooRFinder\SDRuno_PWRSNR.csv
- USB/Serial GPS Puck:
  - Windows: Open Device Manager and find “Prolific USB to Serial COM Port”
    - Example: (COM8)
  - Linux: (TBD)

System Assembly and Setup Cont’d

- Direwolf:
  - Download and unzip to …\flooRFinder folder
  - Edit direwolf.conf with your callsign-SSID
  - Add TBEACON line that includes path to Python Script [name] and beacon period. Suggestion: delay=2, every=2 (2 min delay, beacon every 2 minutes)
  - Configure NMEA GPS COM Port (See USB/Serial GPS Puck info)
- (TBD)

System Assembly and Setup Cont’d

- Windows Task Scheduler:
  - Executable: C:\[python_install_path]\python.exe
  - Argument: .\...\flooRFinder\scripts\read_gps_data.py
  - Trigger: Daily, every 2 minutes (suggested)
- Linux Cron/CronTab
  - (TBD)

Data Gathering: DIREWOLF Process Flow

- SDRUno regularly dumps Date/Time, Freq, Signal, & SNR to CSV file.
- Direwolf TBEACON occurs every n-minutes and triggers:
  - Custom Python Script:
    - Gathers latest SDRUno CSV datum and outputs pre-formatted payload.
  - Direwolf adds custom script output to TBEACON payload.
  - Direwolf adds GPS NMEA data to TBEACON payload.
  - Direwolf records TBEACON payload (and other info) to a local logfile.
  - Direwolf causes APRS beacon, transmitting TBEACON payload:
    - RF (not recommended if APRS and SDR antennas are close in proximity;
    - TCP/Mobile: APRSdroid or other networked or IGATE style mechanism.

Data Gathering: Non-APRS Process Flow

- SDRUno regularly dumps Date/Time, Freq, Signal, & SNR to CSV file.
- Custom Python script scrapes GPS NMEA sentences, to CSV file.
- CHRON/Scheduler trigger custom Python script:
  - Reads SDRUno CSV ‘last entry’
  - Reads GPS NMEA ‘last entry’
  - Creates reportable payload similar to APRS:
    - Date/Time, Latitude DD, Longitude DD, Frequency (Hz), Signal (dB), SNR (dB)
  - Appends payload to logfile

Reporting:
Functional Flow

- User creates a new Google FusionTables,
- User uploads the reporting logfile,
- User assigns 2nd and 3rd columns as coordinates to ‘geo code’,
- User creates a HeatMap using SNR or Signal Strength field,
  - This should show colors indicating relative magnitude of (SNR or Signal).
- User publishes heat map,
- HeatMap Visitor clicks mouse on a location to display logfile data.
- Note: Other possible reporting solutions could be created/used.

Antenna Recommendations

- Resonant antenna for the target band.
  - 6m: Resonant 6m whip.
  - Longer-waves: Use tuner/loaded antenna.
- Non-resonant antennas okay but will skew reported values:
  - Good: Center-loaded/colinear-loaded antenna for the target band,
  - Okay: Center-loaded or colinear-loaded shortened 6m antenna,
  - Okay: Otherwise tuned or bottom/top loaded non-resonant antenna,
  - Not Good: VHF/UHF/SHF (etc) antenna not near target band.
- Most important: Once you select an antenna for use, stick with it:
  - Will ensure consistent results from your system deployment.

Log Schemas

- Custom Python Script Log:
  - GPS NMEA Lat/Lon in both DD and DMS, SDRuno CSV
- SDRuno CSV:
  - VFO Freq (Hz), Date Stamp, VFO Freq MHz, Power (dBm), SNR (dB)
- DIREWOLF ‘LOG .’ log:
  - chan, utime, isotime, source, heard, level, error, dti, name, symbol, latitude, longitude, speed, course, altitude, frequency, offset, tone, system, status, telemetry, comment
  - The ‘comment’ field will contain the latest Custom Python Script Log data

What WAS NOT Done

- Solution Definition and Experimentation with:
  - Hardware TNCs; other ‘soundmodem’ TNCs;
  - SDR products other than RTL’s RSP;
  - Other SDR Software than RTL’s SDRuno.
- Strictly defining a reporting method/mechanism/solution:
  - Many options are available for local and cloud-based reporting;
  - Google Sheets/Fusion are free and a quick way to demonstrate the power of the information gathered by this solution.

References

- Python 3.x Wiki:
- Stack Overflow:
- DIREWOLF:
- Google Fusion:
- Participating hams at CascadiaRadio’s Slack
- Amal G Jose: Sharing his Python script to parse GPS NMEA sentances