



Creating an APRS iGate Using RTL-SDR

Presenter: Chris A. Clark – KK4NXA



RTL-SDR: Review

- SDR Using Compatible DVB-T USB TV Tuner Dongle
 - Realtek RTL2832U Based
 - Receive Only
 - 8 bit Samples at 3.2 MS/s (Max)
 - Frequency Range: 22 MHz – 2200 MHz (Max)
- SDR Software Available on All Major Platforms
 - Windows XP, Vista, 7, 8.x, and 10*
 - Linux (Will Work on Raspberry Pi)
 - Mac OS X (Intel Macs)



APRS iGate: Requirements

- A Recent Computing Platform with Available USB 2.0/3.0 Port
 - x86 / x64 PC
 - Intel Mac
 - Raspberry Pi
- RTL-SDR Dongle and Antenna
- Internet Connection
- Your Favourite Linux Distribution
- About 1 -2 Hours

APRS iGate: Environment Preparation

- Install Linux on Your Host Machine
 - This Presentation Based Upon Ubuntu 14.04
 - Raspbian on Raspberry Pi is Similar with Minor Differences as Noted
- After Installation, Login and Install Distro Updates
- Remove PulseAudio
 - `sudo apt-get remove --purge pulseaudio`
 - `sudo apt-get autoremove`
 - `rm -rf ~/.asoundrc ~/.pulse`
- Prevent the OS From Loading the Default Drivers for RTL-SDR
 - Edit `/etc/modprobe.d/blacklist.conf`
 - Add a blacklist statement for the following drivers:
 - `dvb_usb_rt128xxu`
 - `rtl2832` (`rtl_2832` on Raspbian)
 - `Rtl2839` (`rtl_2830` on Raspbian)

APRS iGate: Environment Preparation

kk4nxa@igate: ~/rtl/rtl-sdr

```
blacklist garmin_gps
```

```
# replaced by asus-laptop (Ubuntu: #184721)
```

```
blacklist asus_acpi
```

```
# low-quality, just noise when being used for sound playback, causes
```

```
# hangs at desktop session start (Ubuntu: #246969)
```

```
blacklist snd_pcsp
```

```
# ugly and loud noise, getting on everyone's nerves; this should be done by a
```

```
# nice pulseaudio bing (Ubuntu: #77010)
```

```
blacklist pcspkr
```

```
# EDAC driver for amd76x clashes with the agp driver preventing the aperture
```

```
# from being initialised (Ubuntu: #297750). Blacklist so that the driver
```

```
# continues to build and is installable for the few cases where its
```

```
# really needed.
```

```
blacklist amd76x_edac
```

```
# BLACKLIST ITEMS BELOW FOR RTL-SDR COMPATIBILITY
```

```
# REMOVE KERNEL MODE DRIVER FOR LIBRTLSDR
```

```
blacklist dvb_usb_rt128xxu
```

```
blacklist rtl_2832
```

```
blacklist rtl_2830
```

```
blacklist rtl2832
```

```
blacklist rtl2830
```

```
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~
```

63,17

Bot

APRS iGate: Environment Preparation

- Reboot the System and Login
- Install the Following Packages or Their Equivalents
 - `sudo apt-get install git build-essential cmake libtool libusb-1.0-0-dev autoconf automake libfftw3-dev qt4-qmake libpulse-dev libx11-dev python-pkg-resources sox`
- Create Directory for Cloned Repositories: `mkdir ~/rtl`
- Clone Required Git Repositories
 - `sudo git clone git://git.osmocom.org/rtl-sdr.git`
 - `sudo git clone https://github.com/asdil12/kalibrate-rtl.git`
 - `sudo git clone https://github.com/EliasOenal/multimonNG.git`
 - `sudo git clone https://github.com/asdil12/pymultimonaprs.git`

APRS iGate: RTL-SDR Driver & Software

Compile and Install the New RTL-SDR Driver and Software
Downloaded from OSMOCOM:

```
cd ~/rtl/rtl-sdr
mkdir build
cd build
cmake .. -DINSTALL_UDEV_RULES=ON
make
sudo make install
sudo ldconfig
```

After Installation, Connect RTL-SDR Dongle to Test

APRS iGate: Testing Dongle with RTL-SDR

Running `rtl_test` at the CLI should provide this output:

```
kk4nxa@igate: ~/rtl/rtl-sdr
Found Rafael Micro R820T tuner
Supported gain values (29): 0.0 0.9 1.4 2.7 3.7 7.7 8.7 12.5 14.4 15.7 16.6 19.7 20.7 22.9 25.4 28.0 29.7 32.8 33.8 36.4 37.2 38.6 40.2 42.1 43.4 43.9 44.5 48.0 49.6
Sampling at 2048000 S/s.

Info: This tool will continuously read from the device, and report if
samples get lost. If you observe no further output, everything is fine.

Reading samples in async mode...

^CSignal caught, exiting!

User cancel, exiting...
Samples per million lost (minimum): 0
kk4nxa@igate:~/rtl/rtl-sdr$ rtl_fm -M wbfm -f 90.7M | play -r 32k -t raw -e s -b 16 -c 1 -V1 -
Found 1 device(s):
 0: Realtek, RTL2838UHIDIR, SN: 00000001

Using device 0: Generic RTL2832U OEM
Found Rafael Micro R820T tuner

-: (raw)

Encoding: Signed PCM
Channels: 1 @ 16-bit
Samplerate: 32000Hz
Replaygain: off
Duration: unknown

In:0.00% 00:00:00.00 [00:00:00.00] Out:0 [ | ] Clip:0 Tuner gain set to automatic.
Tuned to 90971000 Hz.
Oversampling input by: 6x.
Oversampling output by: 1x.
Buffer size: 8.03ms
Exact sample rate is: 1020000.026345 Hz
Sampling at 1020000 S/s.
Output at 170000 Hz.
In:0.00% 00:00:21.50 [00:00:00.00] Out:1.02M [ ====|==== ] Clip:0
```

Next, Test Audio Playback with `rtl_fm`:

```
rtl_fm -M wbfm -f 90.7M | play -r 32k -t raw -e s -b 16 -c 1 -V1
```

APRS iGate: Testing Dongle with RTL-SDR

If Either Command Provides Output Similar to the Below Image, You Have Not Prevented the Default Kernel mode Driver from Loading by Including It in the Blacklist

```
kk4nxa@igate: ~/rtl/rtl-sdr/build
Processing triggers for mime-support (3.54ubuntu1.1) ...
Processing triggers for man-db (2.6.7.1-1ubuntu1) ...
Setting up libsox2:amd64 (14.4.1-3ubuntu1) ...
Setting up libsox-fmt-alsa:amd64 (14.4.1-3ubuntu1) ...
Setting up libsox-fmt-base:amd64 (14.4.1-3ubuntu1) ...
Setting up sox (14.4.1-3ubuntu1) ...
Processing triggers for libc-bin (2.19-0ubuntu6.5) ...
kk4nxa@igate:~/rtl/rtl-sdr/build$ rtl_test
Found 1 device(s):
 0: Realtek, RTL2838UHIDIR, SN: 00000001

Using device 0: Generic RTL2832U OEM

Kernel driver is active, or device is claimed by second instance of librtlsdr.
In the first case, please either detach or blacklist the kernel module
(dvb_usb_rtl28xxu), or enable automatic detaching at compile time.

usb_claim_interface error -6
Failed to open rtlsdr device #0.
kk4nxa@igate:~/rtl/rtl-sdr/build$ rtl_fm -M wbfm -f 91.7M | play -r 32k -t raw -e s -b 16 -c 1 -V1 -
Found 1 device(s):
 0: Realtek, RTL2838UHIDIR, SN: 00000001

Using device 0: Generic RTL2832U OEM

Kernel driver is active, or device is claimed by second instance of librtlsdr.
In the first case, please either detach or blacklist the kernel module
(dvb_usb_rtl28xxu), or enable automatic detaching at compile time.

usb_claim_interface error -6
Failed to open rtlsdr device #0.

-: (raw)

  Encoding: Signed PCM
  Channels: 1 @ 16-bit
 Samplerate: 32000Hz
 Replaygain: off
  Duration: unknown

In:0.00% 00:00:00.00 [00:00:00.00] Out:0 [ ] Clip:0
Done.
kk4nxa@igate:~/rtl/rtl-sdr/build$
```

APRS iGate: Frequency Calibration

Due to the low-cost components used, all RTL-SDR dongles have a slight frequency offset. The Kalibrate-RTL software can detect this offset and display it.

Compile and Install the Kalibrate-RTL Software Downloaded from GitHub:

```
cd ~/rtl/kalibrate-rtl
!(On Raspberry Pi Only)! git checkout arm_memory
./bootstrap
./configure
make
sudo make install
```

After installation, run Kalibrate-RTL to discover the frequency offset. You will first need to identify a strong cellular channel (GSM900 is good):

```
kal -s GSM900
```

It will then use the clock signal on the known station frequency to calculate the frequency offset of the dongle:

```
kal -c 13
```

APRS iGate: Frequency Calibration

```
kk4nxa@igate: ~/rtl/pymultimonaprs
Found Rafael Micro R820T tuner
Exact sample rate is: 270833.002142 Hz
kal: Scanning for GSM-900 base stations.
GSM-900:
    chan: 13 (937.6MHz + 15.288kHz) power: 18423.11
^C
kk4nxa@igate:~/rtl/pymultimonaprs$ kal -c 13 -e
kal: option requires an argument -- 'e'
kalibrate v0.4.1-rtl, Copyright (c) 2010, Joshua Lackey
modified for use with rtl-sdr devices, Copyright (c) 2012, Steve Markgraf
Usage:
    GSM Base Station Scan:
        kal <-s band indicator> [options]

    Clock Offset Calculation:
        kal <-f frequency | -c channel> [options]

Where options are:
    -s    band to scan (GSM850, GSM-R, GSM900, EGSM, DCS, PCS)
    -f    frequency of nearby GSM base station
    -c    channel of nearby GSM base station
    -b    band indicator (GSM850, GSM-R, GSM900, EGSM, DCS, PCS)
    -g    gain in dB
    -d    rtl-sdr device index
    -e    initial frequency error in ppm
    -v    verbose
    -D    enable debug messages
    -h    help
kk4nxa@igate:~/rtl/pymultimonaprs$ kal -c 13
Found 1 device(s):
    0: Generic RTL2832U OEM

Using device 0: Generic RTL2832U OEM
Found Rafael Micro R820T tuner
Exact sample rate is: 270833.002142 Hz
kal: Calculating clock frequency offset.
Using GSM-900 channel 13 (937.6MHz)
```

APRS iGate: Decoding APRS

Use the multimonNG software downloaded from GitHub to decode APRS data packets. The iGate software will call multimonNG for this purpose.

```
cd ~/rtl/multimonNG
mkdir build
cd build
qmake ../multimon-ng.pro
make
sudo make install
```

You can now pipe the audio received from rtl_fm into multimon-ng and view decoded APRS packets:

```
rtl_fm -f 144.390M -s 22050 [-p offset] | multimon-ng -a AFSK1200 -A -t raw
```

APRS iGate: Decoding APRS

```
kk4nxa@igate: ~/rtl
kk4nxa@igate:~/rtl$ rtl_fm -f 144.390M -s 22050 - | multimon-ng -a AFSK1200 -A -t raw -
Found 1 device(s):
multimon-ng (C) 1996/1997 by Tom Sailer HB9JNX/AE4WA
              (C) 2012-2014 by Elias Oenal
available demodulators: POCSSAG512 POCSSAG1200 POCSSAG2400 EAS UFSK1200 CLIPFSK FMSFSK AFSK1200 AFSK2400 AFSK2400_2 AFSK2400_3 HAPN
4800 FSK9600 DTMF ZVEI1 ZVEI2 ZVEI3 DZVEI PZVEI EEA EIA CCIR MORSE_CW DUMPCSV SCOPE
Enabled demodulators: AFSK1200
  0: Realtek, RTL2838UHIDIR, SN: 00000001

Using device 0: Generic RTL2832U OEM
Found Rafael Micro R820T tuner
Tuner gain set to automatic.
Tuned to 144643575 Hz.
Oversampling input by: 46x.
Oversampling output by: 1x.
Buffer size: 8.08ms
Exact sample rate is: 1014300.020041 Hz
Sampling at 1014300 S/s.
Output at 22050 Hz.
```

APRS iGate: Install iGate Software

Use the pymultimonaprs software downloaded from GitHub to forward decoded APRS packets to the APRS-IS network:

```
cd ~/rtl/pymultimonaprs
sudo python setup.py build
sudo python setup.py install
```

After installation, generate a APRS-IS passcode for your callsign:

```
./keygen.py KK4NXA
Key for KK4NXA: XXXXXX
```

Next, modify the pymultimonaprs.json file for your callsign, passcode, coordinates, and desired APRS-IS Tier 2 Gateway:

```
sudo [vim/nano/pico] /etc/pymultimonaprs.json
```

APRS iGate: Install iGate Software

```
kk4nxa@igate: ~/rtl/pymultimonaprs
{
  "callsign": "KK4NXA",
  "passcode": " ",
  "gateway": "us.aprs2.net:14580",
  "append_callsign": true,
  "source": "rtl",
  "rtl": {
    "freq": 144.390,
    "ppm": 0,
    "gain": 0,
    "offset_tuning": false,
    "device_index": 0
  },
  "alsa": {
    "device": "default"
  },
  "beacon": {
    "lat": 33.505041,
    "lng": -81.733505,
    "table": "/",
    "symbol": "&",
    "comment": "PyMultimonAPRS iGate",
    "status": {
      "text": "Running on Linux with RTL-SDR dongle",
      "file": false
    },
    "weather": false,
    "send_every": 300,
    "ambiguity": 0
  }
}
~
~
~
~
~
-- INSERT --
```

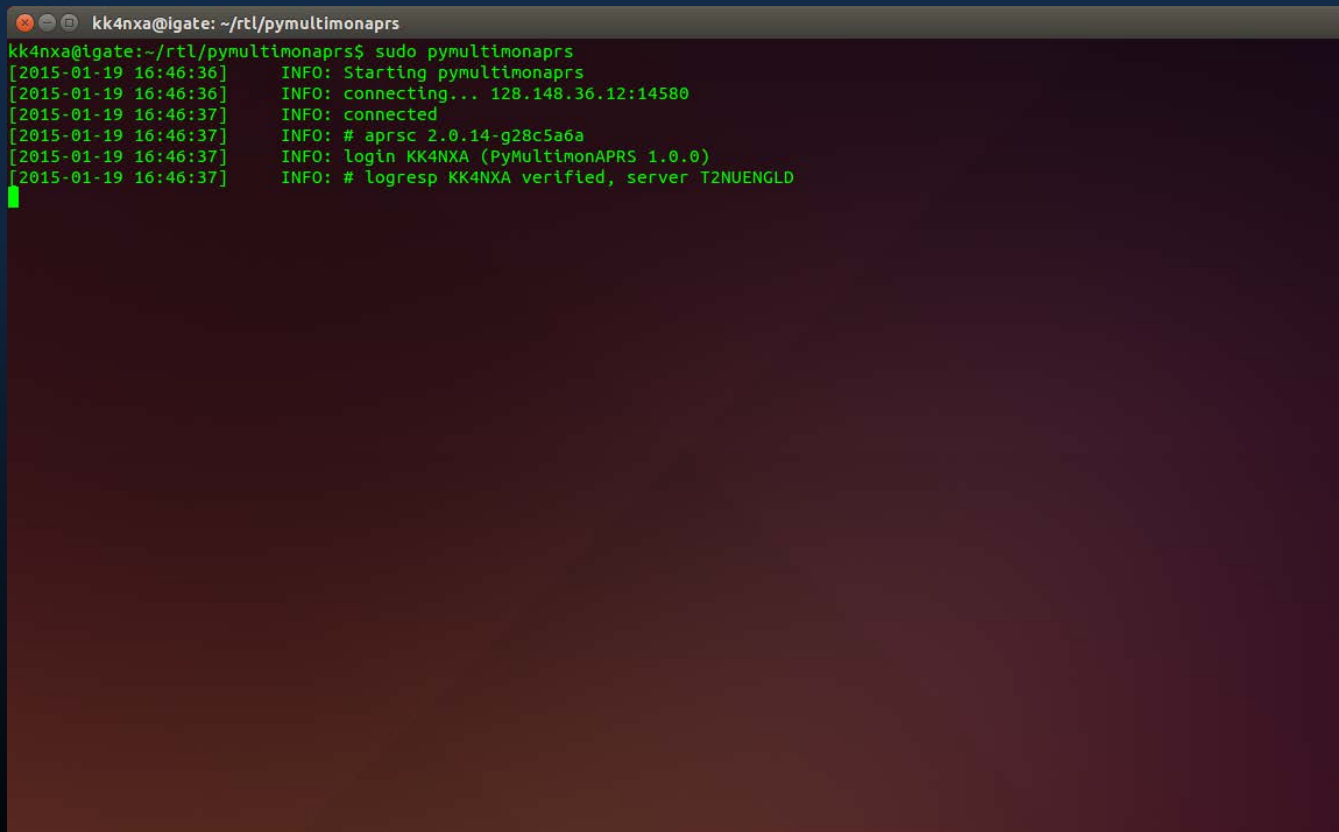
24,42-63 All

APRS iGate: Testing iGate Software

After modifying the `/etc/pymultimonaprs.json` file, we can run the software:

```
sudo pymultimonaprs
```

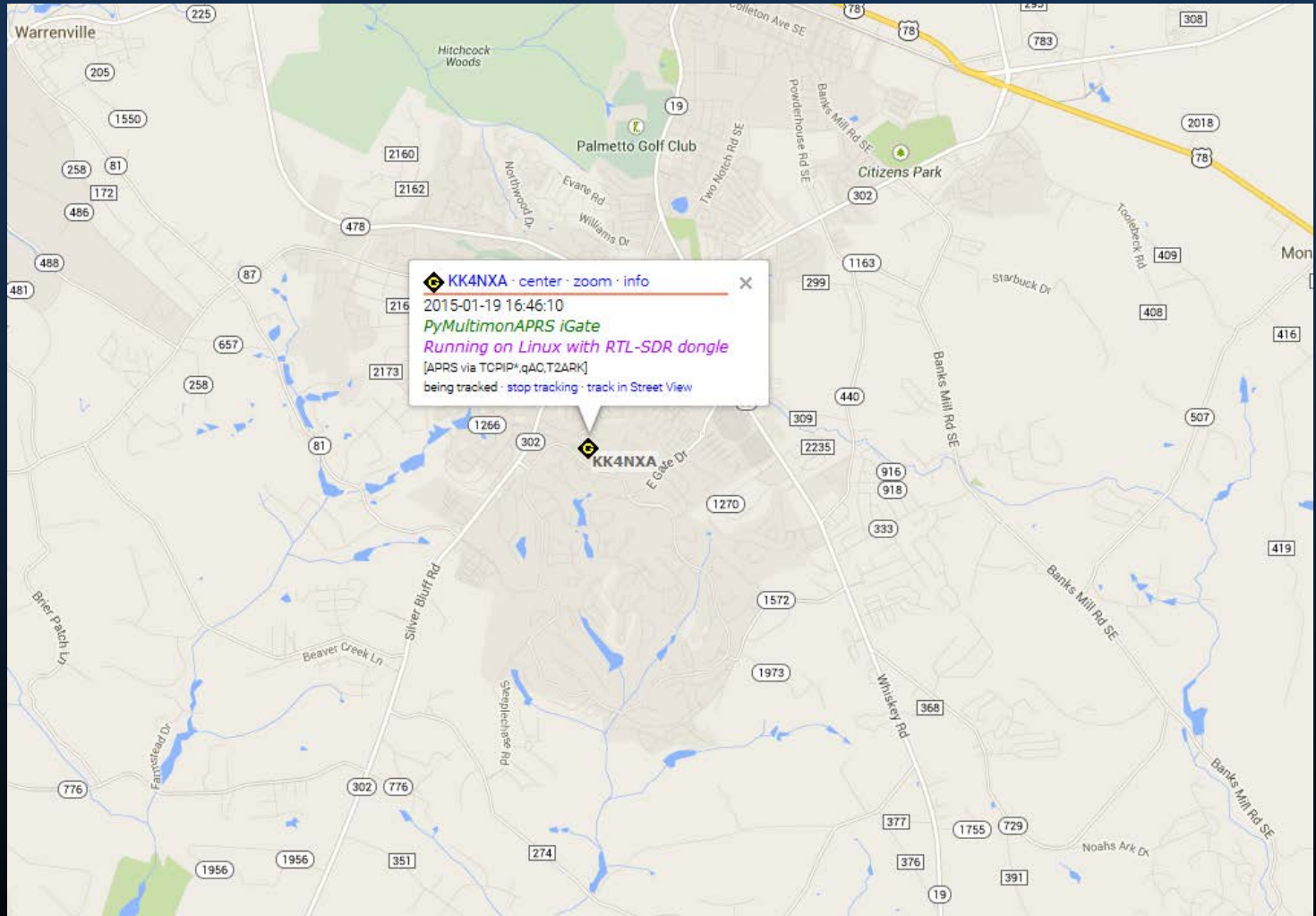
A correct execution will look something like this:



```
kk4nxa@igate: ~/rtl/pymultimonaprs
kk4nxa@igate:~/rtl/pymultimonaprs$ sudo pymultimonaprs
[2015-01-19 16:46:36]      INFO: Starting pymultimonaprs
[2015-01-19 16:46:36]      INFO: connecting... 128.148.36.12:14580
[2015-01-19 16:46:37]      INFO: connected
[2015-01-19 16:46:37]      INFO: # aprsc 2.0.14-g28c5a6a
[2015-01-19 16:46:37]      INFO: login KK4NXA (PyMultimonAPRS 1.0.0)
[2015-01-19 16:46:37]      INFO: # logresp KK4NXA verified, server T2NUENGLD
```

APRS iGate: Testing iGate Software

Success:



APRS iGate: Final Steps

If this is a permanent iGate installation, you'll want to create a startup script:

```
sudo [vim/nano/pico] /etc/init.d/pymultimonaprs
```

Paste:

```
#!/bin/sh
### BEGIN INIT INFO
# Provides: pymultimonaprs
# Required-Start: $all
# Required-Stop: $syslog
# Default-Start: 2 3 4 5
# Default-Stop: 0 1 6
# Short-Description: start/stop of pymultimonaprs
### END INIT INFO
case "$1" in
start)
sudo pymultimonaprs --syslog &
;;
stop)
sudo killall pymultimonaprs
;;
*)
echo "Usage: /etc/init.d/pymultimonaprs {start|stop}"
exit 1
;;
esac
exit 0
```

APRS iGate: Final Steps

Enable Your Startup Script:

```
sudo chmod +x /etc/init.d/pymultimonaprs  
sudo update-rc.d pymultimonaprs defaults  
  
sudo /etc/init.d/pymultimonaprs start
```

Congratulations! You're Done!

Demonstration

Questions?

APRS iGate: Final Thoughts

- Quickly, Cheaply Deploy an iGate to Extend APRS Coverage
 - Permanently Provide Coverage to A Poorly Covered Area
 - Deploy a Mobile or Portable Solution for Coverage During an Event
 - Deploy a Portable Solution on the Road
 - Deploy a Portable Solution in a HAB
- Deploy on Raspberry Pi for Portable iGate APRS Coverage
- This Should Also Work on Other x86 Embedded Systems

Thank You