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)

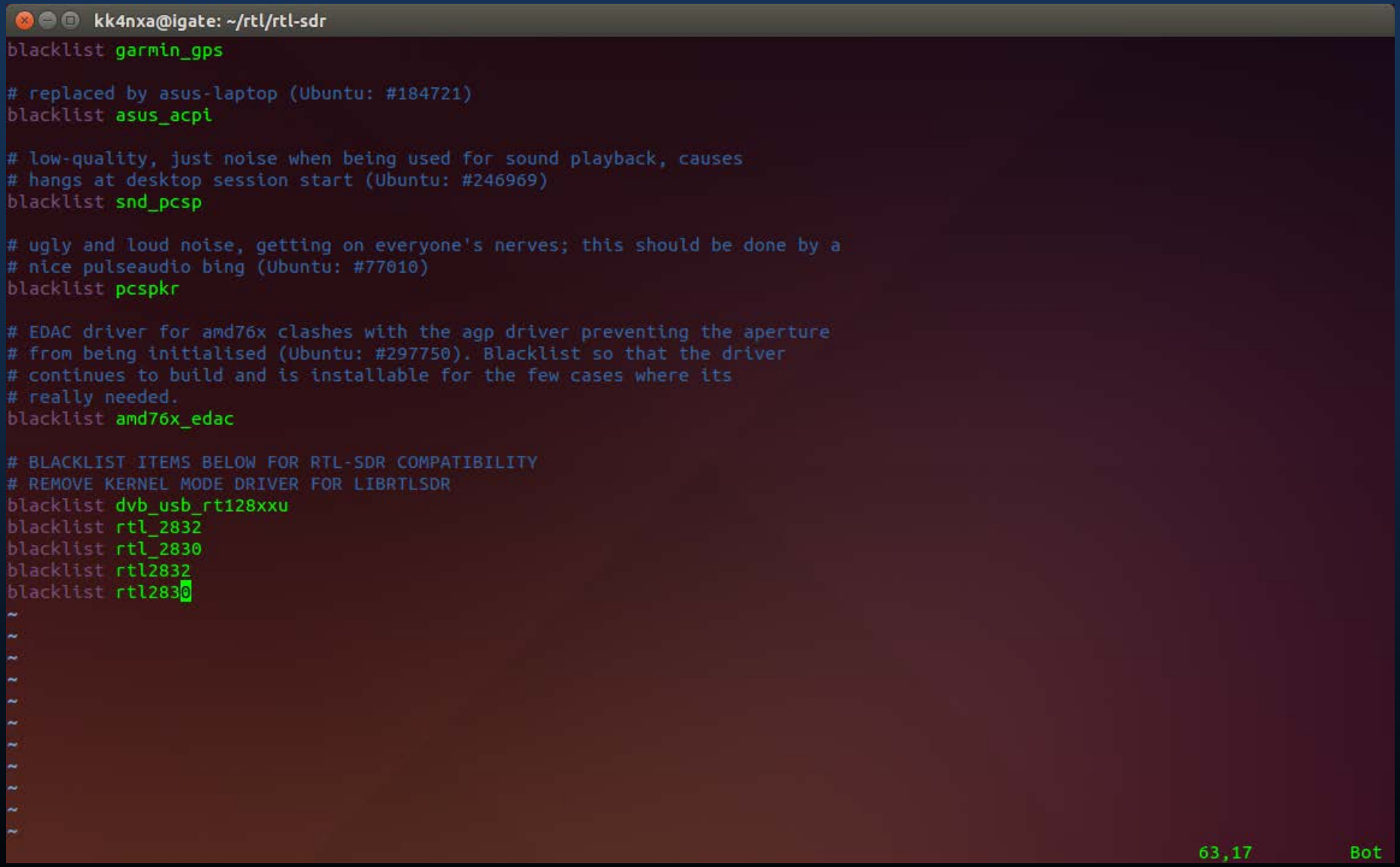
APRS iGate: Requirements

- 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



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

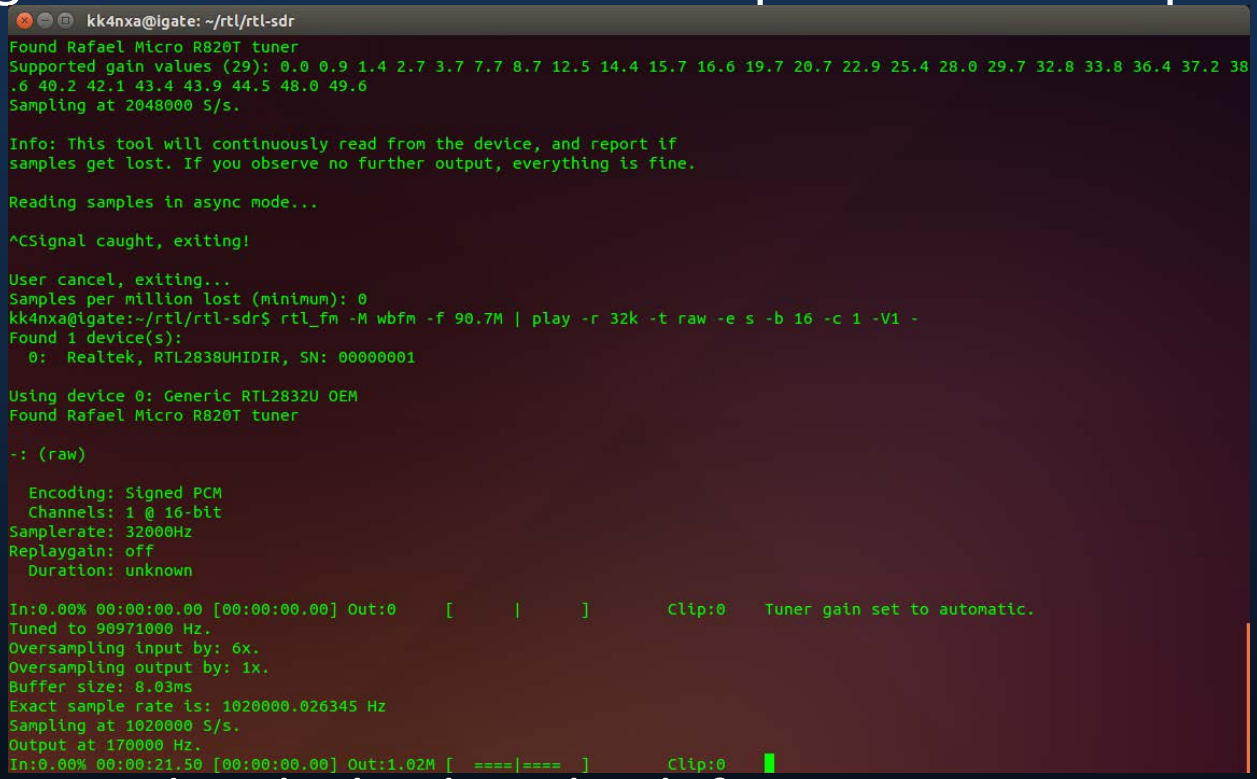
Compile and Install the New RTL-SDR Driver and Software Downloaded from OSMOCOM:

```
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:

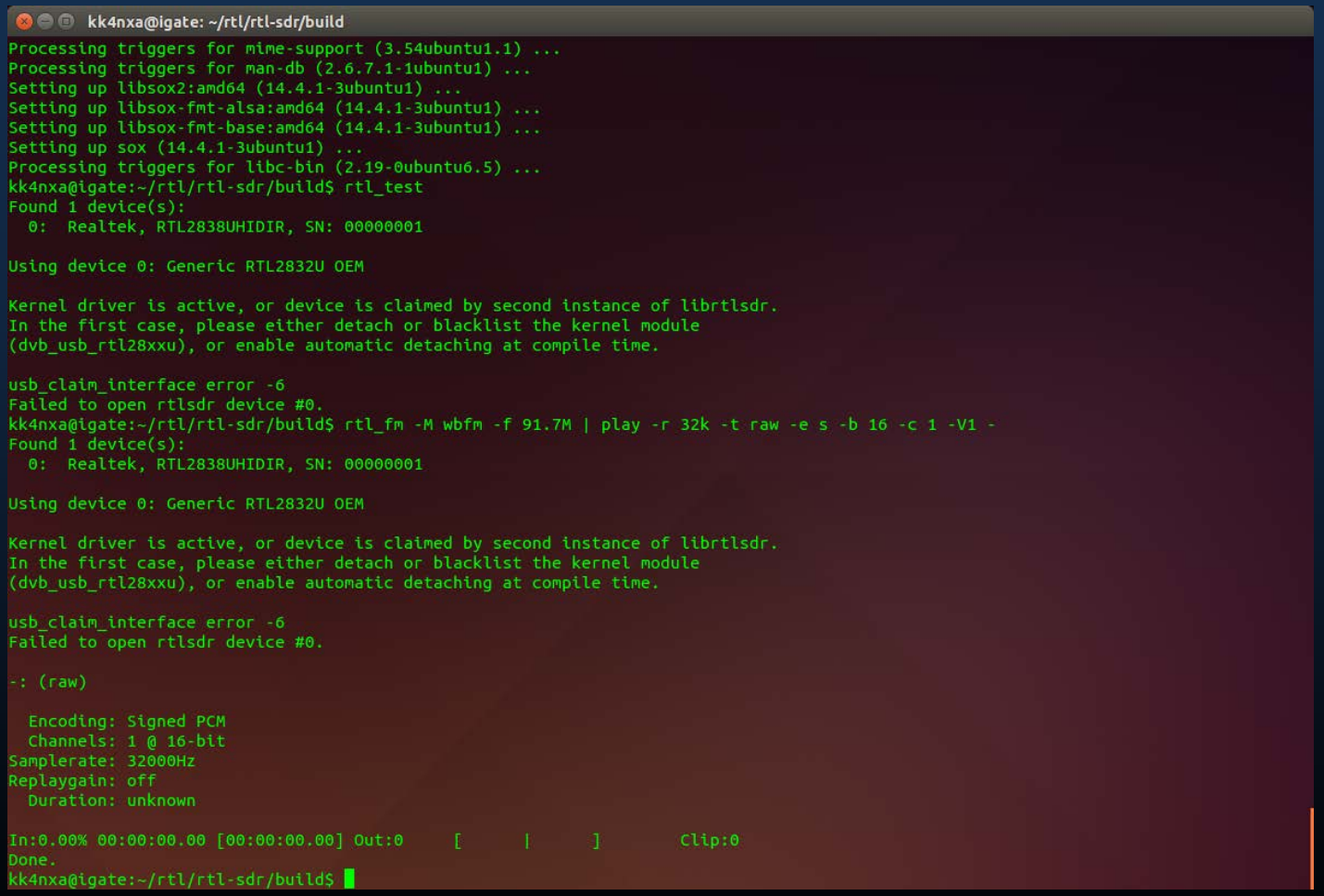


Next, Test Audio Playback with rtl_fm:

```
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



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
```

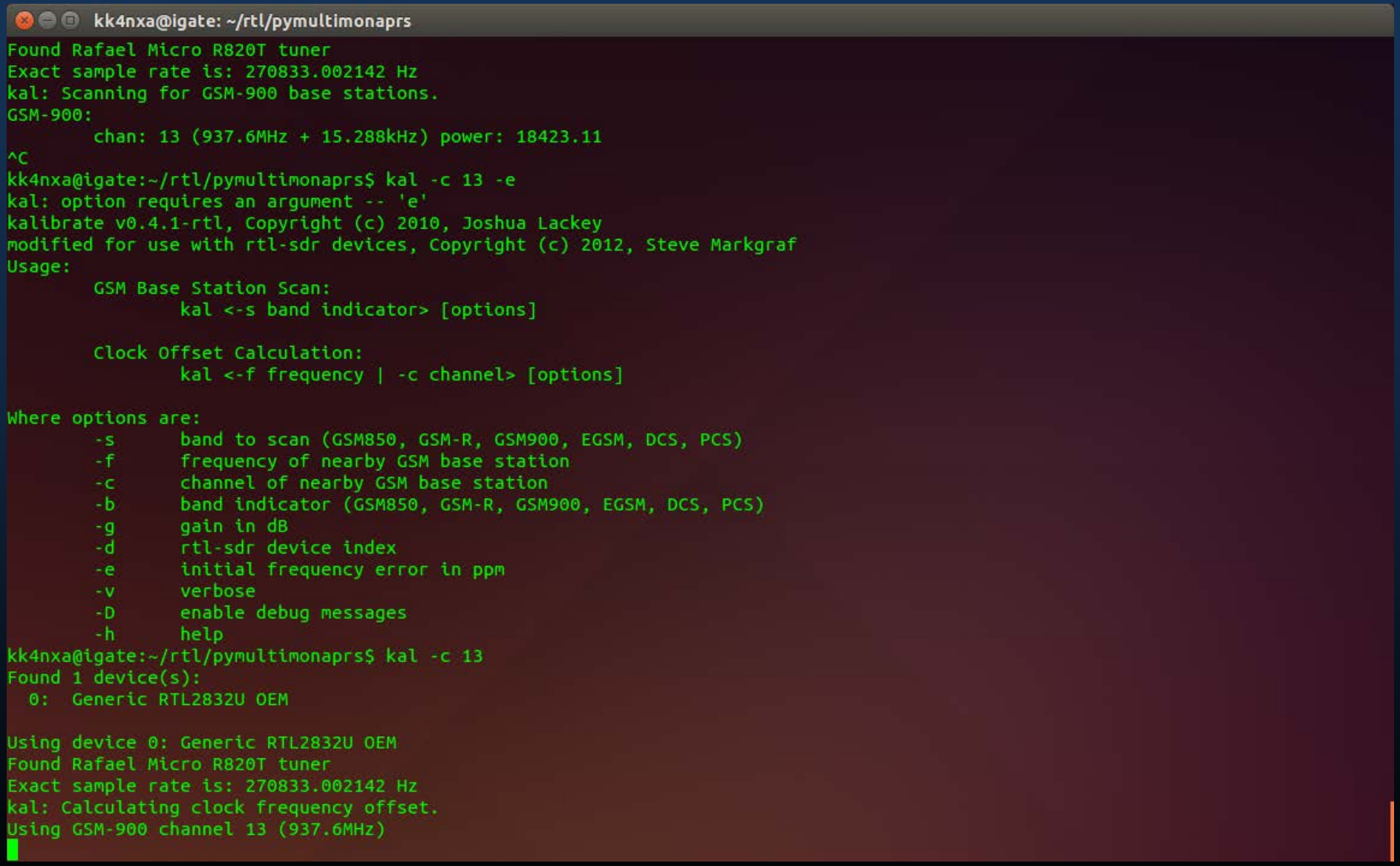
After installation, run Kalibrate-RTL to discover the frequency offset. You will first need to identify a strong cellular channel (GSM900 is good):

```
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



APRS iGate: Decoding APRS

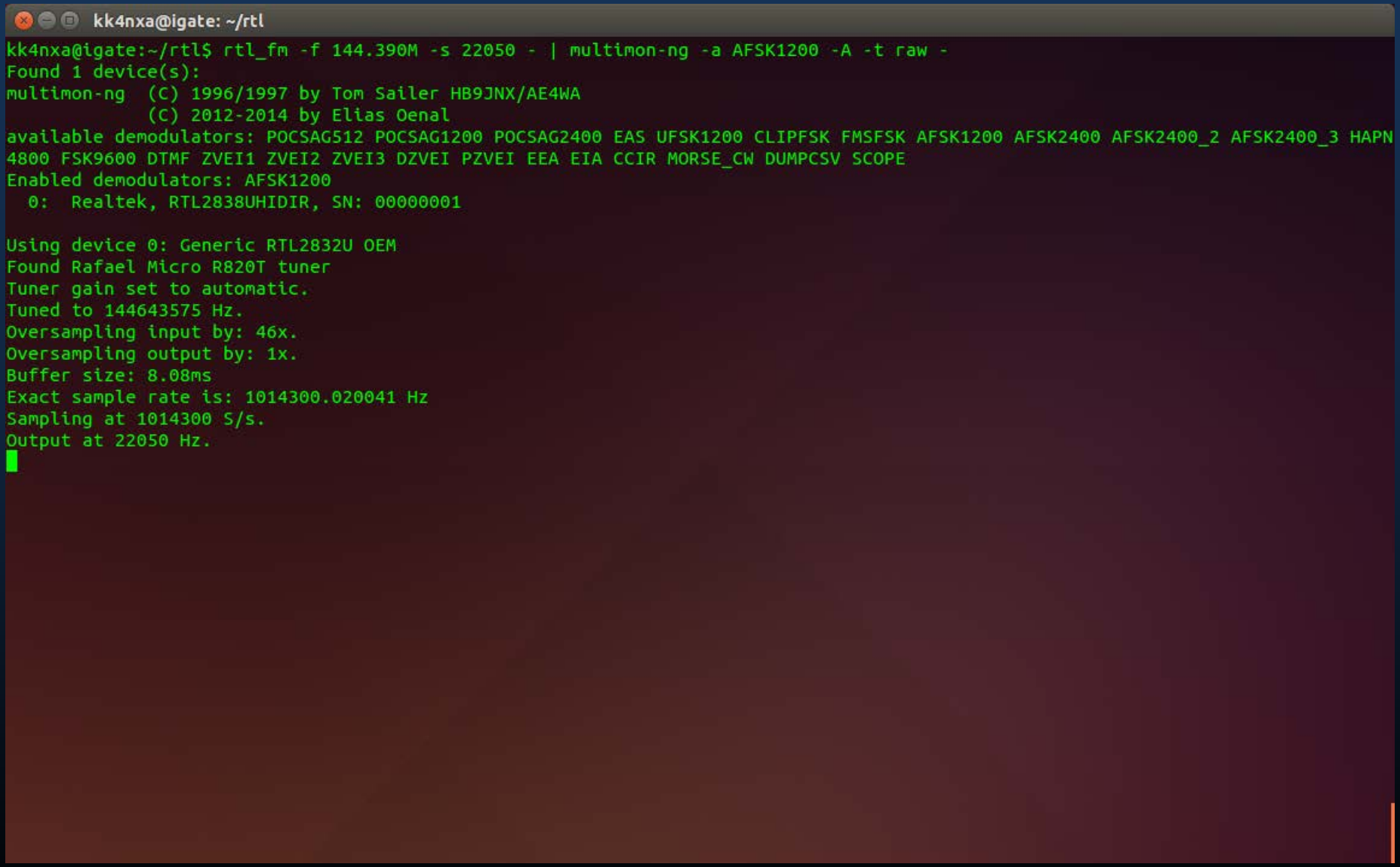
Use the multimonNG software downloaded from GitHub to decode APRS data packets. The iGate software will call multimonNG for this purpose.

```
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



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

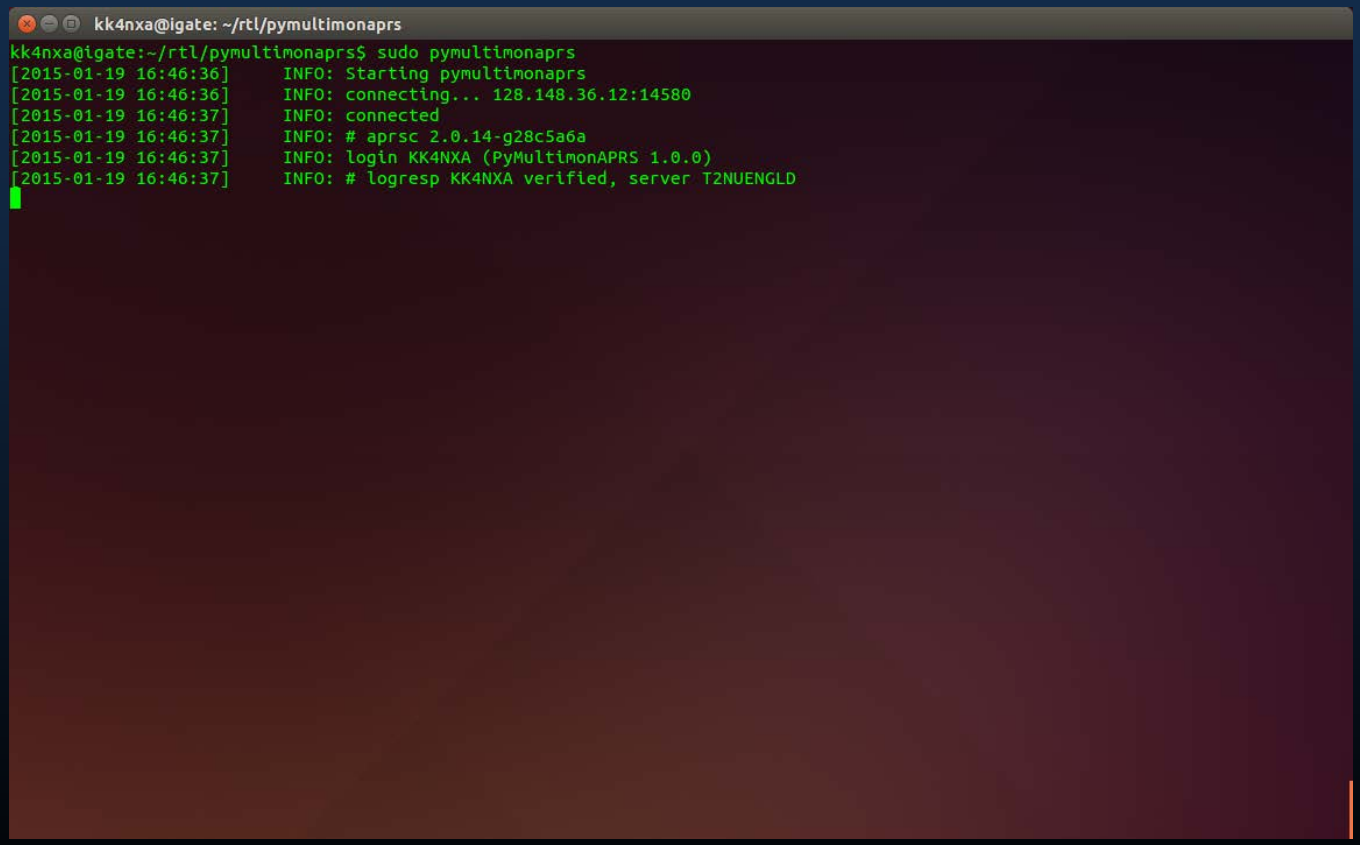
```
{
        "callsign": "KK4NXA",
        "passcode": "[redacted]",
        "gateway": "us.aprs2.net:14580",
        "append_callsign": true,
        "source": "rtl",
        "rtl": {
                "freq": 144.390,
                "ppm": 0,
                "gain": 0,
                "offset_tuning": 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:

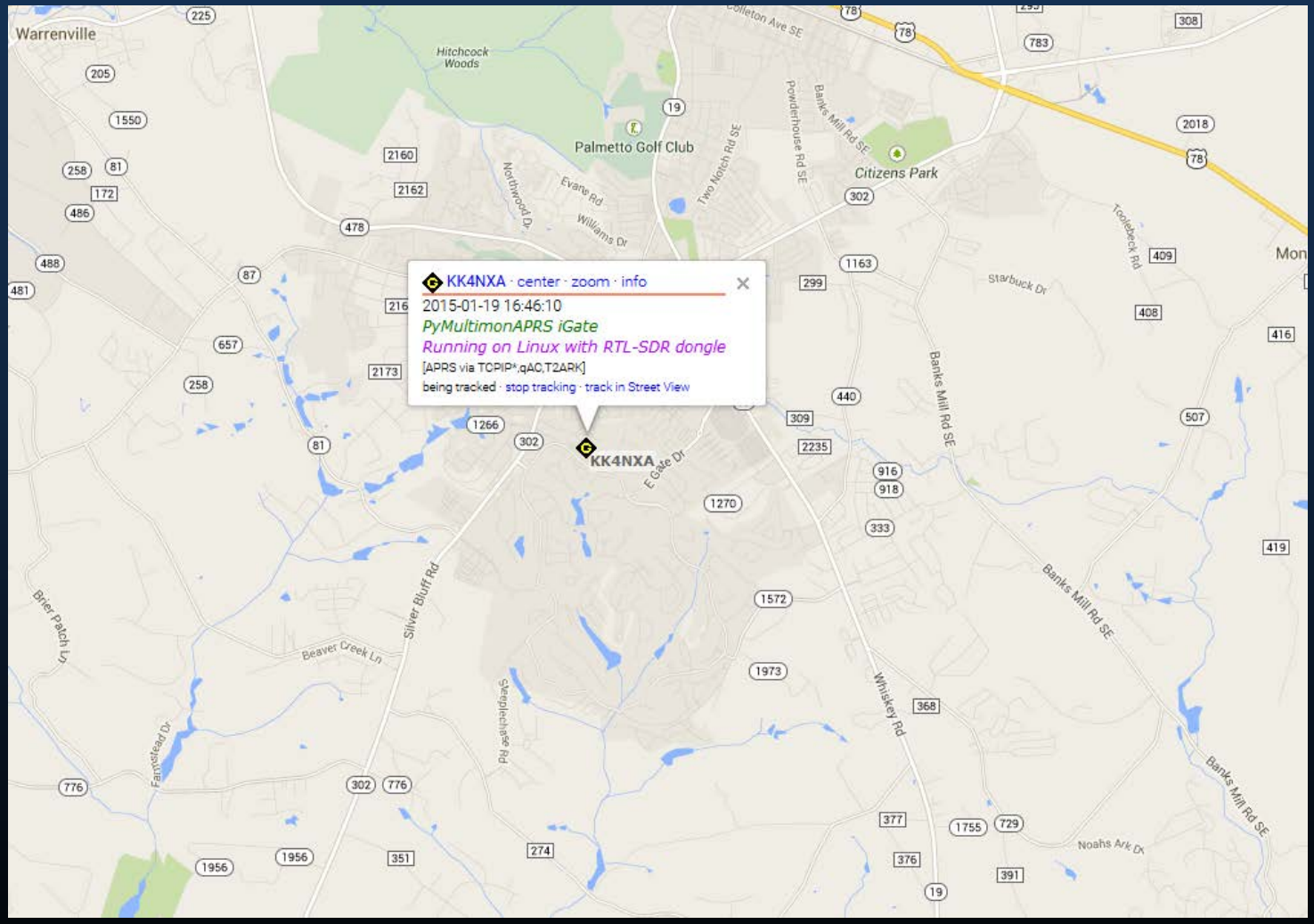
```
sudo pymultimonaprs
```

A correct execution will look something like this:



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