iGate / Digipeater APRX on a RASPBERRY PI using Soundmodem

Thursday, 19 February 2015
By PA0ESH
For questions sent me an email at pa0esh at amsat dot org
Includes now how to compile aprx from trunk – in orderto get the latest version

This document describes step by step how to install an iGate / Digipeater on a Raspberry PI, using soundmodem as the TNC.

It is noted that for the time being, you have to twiddle a bit around with the TX delay settings of the soundmodem,, because I am still implementing the hardware to make the TX go ON/OFF so for the time being , the TX should be able to work with VOX.

I have updated the install script for aprs on a Raspberry. It now gives you the selection of installing:

- Soundmodem incl. testing and adjusting of the audio
- aprx
- dixprs
- xastir
- gpsd

Create a fresh debian wheezy 4Gb card and get in script here. Put it in /usr/src , chmod 755 the file and then run it: /usr/src/aprx.sh

sudo apt-get update && sudo apt-get -y dist-upgrade && cd /usr/src && wget http://www.pa0esh.nl/aprs/rpi/aprx.sh && chmod 755 aprx.sh && ./aprx.sh

However, the step-by step method teaches a bit more about what is going on.

Step 1

Create a fresh image of Raspbian “wheezy”, which you can download from here and where they also provide many ilnks to how to do this….
Personally I use a little programme called PiWriter.
http://sourceforge.net/projects/piwriter/

Since I do not have a HDMI screen for the Raspberry, I use the SSH method, either direct from a Linux machine or through putty on windows. Take your pick.

Find the IP address of the Raspberry after it has booted.
SSH into the machine using pi as the username – The password is raspberry
Execute $ **sudo su**

Then change the su root password by executing ***passwd***
So now we have both a user (pi) as well as a root account under control.

Step 2

Update the Raspberry to the latest status
$ sudo apt-get update
Now upgrade the Raspberry
$ sudo apt-get upgrade

then, edit the file /boot/cmdline.txt and add at the following :
dwc_otg.speed=1
This brings the usb port back to usb1.0 but that is fine for the audio, even better I believe…

Since we are going to install aprx , we do not require much more software except required for the ax25 soundmodem stack.

Some help on ax25 can be found here:
http://www.xastir.org/wiki/HowTo:AX.25
http://f6bvp.org/
http://71.49.9.157:800/docs/Raspberry.html

Step 3

Soundmodem and soundmodem tools installation

$ **apt-get install soundmodem ax25-tools ax25-apps libax25-dev screen**

So now a bit of hardware

I use a very cheap soundcard - from eBay…, which looked like this and did not cost much. But that may also create problems. So look around for USB sound modules, and be prepared to be disappointed that you may have to buy another one.
In my case one worked, and one did not…

Connect the USB dongle to the Raspberry and check what you have got.

Note: I would strongly advice using a powered USB hub since the current it draws from the Raspberry is too much…

$ lsusb

```
Bus 001 Device 002: ID 0424:9512 Standard Microsystems Corp.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 003: ID 0424:ec00 Standard Microsystems Corp.
Bus 001 Device 004: ID 148f:5370 Ralink Technology, Corp. RT5370 Wireless Adapter
Bus 001 Device 005: ID 1a40:0101 Terminus Technology Inc. 4-Port HUB
Bus 001 Device 006: ID 0d8c:000e C-Media Electronics, Inc. Audio Adapter (Planet UP-100, Genius G-Talk)
Bus 001 Device 007: ID 0403:6001 Future Technology Devices International, Ltd FT232 USB-Serial (UART) IC
Bus 001 Device 008: ID 10c4:ea60 Cygnal Integrated Products, Inc. CP210x UART Bridge / myAVR mySmartUSB light
```

The green line showed up as the type of sound module and there is plenty of info about this little machine on the net.

Raspbian "wheezy" comes with the alsa sound system pre installed so let's see what we have…

$ aplay –l

```
**** List of PLAYBACK Hardware Devices ****
card 0: ALSA [bcm2835 ALSA], device 0: bcm2835 ALSA [bcm2835 ALSA]
  Subdevices: 8/8
  Subdevice #0: subdevice #0
  Subdevice #1: subdevice #1
  Subdevice #2: subdevice #2
  Subdevice #3: subdevice #3
  Subdevice #4: subdevice #4
  Subdevice #5: subdevice #5
  Subdevice #6: subdevice #6
  Subdevice #7: subdevice #7
card 1: Device [Generic USB Audio Device], device 0: USB Audio [USB Audio]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
```

Hmmm, the first module is the build in sound module, and our USB dongle is listed as number 2.
However, we wish to make this the primary sound device so we nee to tweek a little around.

We have to edit the /etc/modprobe.d/alsa-base.conf file to make the USB sound dongle come first.
$ nano /etc/modprobe.d/alsa-base.conf gives us:

```
# autoloader aliases
install sound-slot-0 /sbin/modprobe snd-card-0
install sound-slot-1 /sbin/modprobe snd-card-1
install sound-slot-2 /sbin/modprobe snd-card-2
install sound-slot-3 /sbin/modprobe snd-card-3
install sound-slot-4 /sbin/modprobe snd-card-4
install sound-slot-5 /sbin/modprobe snd-card-5
```

```
install sound-slot-6 /sbin/modprobe snd-card-6
install sound-slot-7 /sbin/modprobe snd-card-7
# Cause optional modules to be loaded above generic modules
install snd /sbin/modprobe --ignore-install snd && { /sbin/modprobe --quiet snd-ioctl32 ; /sbin/modprobe --quiet snd-seq ; : ; }
install snd-rawmidi /sbin/modprobe --ignore-install snd-rawmidi && { /sbin/modprobe --quiet snd-seq-midi ; : ; }
install snd-emu10k1 /sbin/modprobe --ignore-install snd-emu10k1 && { /sbin/modprobe --quiet snd-emu10k1-synth ; : ; }
# Keep snd-pcsp from beeing loaded as first soundcard
options snd-pcsp index=-2
# Keep snd-usb-audio from beeing loaded as first soundcard
options snd-usb-audio index=-2
# Prevent abnormal drivers from grabbing index 0
options bt87x index=-2
options cx88_alsa index=-2
options snd-atiixp-modem index=-2
options snd-intel8x0m index=-2
options snd-via82xx-modem index=-2
```

Now replace the lines

```
# Keep snd-pcsp from beeing loaded as first soundcard
options snd-pcsp index=-2
# Keep snd-usb-audio from beeing loaded as first soundcard
options snd-usb-audio index=-2
# Prevent abnormal drivers from grabbing index 0
```

with

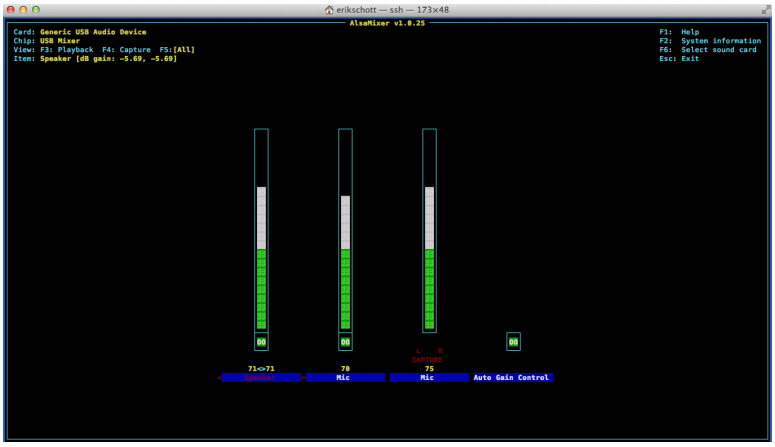
```
options snd slots=snd_bcm2835,snd_usb_audio
options snd_usb_audio index=0
options snd_bcm2835 index=2
```

and reboot

Now type aplay –l and you should see this

$ aplay -l

**** List of PLAYBACK Hardware Devices ****
card 0: Device [Generic USB Audio Device], device 0: USB Audio [USB Audio]
  Subdevices: 1/1
  Subdevice #0: subdevice #0



Now we check the sound system:

Start alsamixer (**$ alsamixer**), and unmute the sound channels (use F5 and F6 to see more of the controls, the M-key mutes / unmutes a channel and you

with by pressing the left-arrow / right arrow keys.

Connect a speaker headphone as well as a microphone for the final tests

$ **speaker-test -c 2** gets you noise at the headphone switching from left to right.
Next we check if the sound modules are correctly loaded

$ **cat /etc/modules**

```
# /etc/modules: kernel modules to load at boot time.
#
# This file contains the names of kernel modules that should be loaded
# at boot time, one per line. Lines beginning with "#" are ignored.
# Parameters can be specified after the module name.

snd-bcm2835
```

Here we see that the USB module is **not** loaded so we change the modules file

$ nano /etc/modules
and add the USB module snd-usb-audio

Note if you are not using the on board sound module at all. You can also comment it out here

Anyway, after saving the command cat /etc/module should give us the following:
$ cat /etc/modules

```
# /etc/modules: kernel modules to load at boot time.
#
# This file contains the names of kernel modules that should be loaded
# at boot time, one per line. Lines beginning with "#" are ignored.
# Parameters can be specified after the module name.

snd-bcm2835
snd-usb-audio
```

The next file we have to change completely for this usb sound module is /etc/asound.conf

If it is not there create it , otherwhile replace it with the following content :
$ **nano /etc/asound.conf**

```
pcm.dmixer {
  type dmix
```

```
  ipc_key 1024
  slave {
    pcm "hw:0,0"
    period_time 0
    period_size 1024
    buffer_size 8192
  rate 48000
  }
  bindings {
    0 0
    1 1
  }
}
pcm.asymed {
    type asym
    playback.pcm "dmixer"
    capture.pcm "hw:0,0"
}
pcm.dsp0 {
  type plug
  slave.pcm "asymed"
}
pcm.!default {
    type plug
    slave.pcm "asymed"
}
pcm.default {
  type plug
  slave.pcm "asymed"
}
ctl.mixer0 {
  type hw
  card 0
}
```

Now reboot again, start alsamixer, make sure that ALL controls are at approx. 80%, also the capture . … press F5 to get there… and press exit

Now record a short soundfile and play back.
Record a file : **$ arecord -vv testsound.wav** (Ctrl-C to stop)
Play back that same file : **$ aplay testsound.wav**

If all is correct, you should hear back whatever you recorded. Fiddle around with the volume controls for the best result.

You could already connect your radio audio out to the mike input and check volumes.

Step 4

The soundmodem configuration file

There are two ways to configure the soundmodem, one using the desktop utility – for which you have work with X11 on SSH or connect a screen to the Raspberry
I prefer to enter the file directly into it's directy /etc/ax25.

$ nano /etc/ax25/soundmodem.conf

And replace it's contents with this

```
<?xml version="1.0"?>
<modem>
<configuration name="AX25">
<chaccess txdelay="250" slottime="100" ppersist="40" fulldup="0"
txtail="100"/>
<audio type="alsa" device="plughw:0,0" halfdup="1"
capturechannelmode="Mono"/>
<ptt file="none" hamlib_model="" hamlib_params="" gpio="0"/>
<channel name="sm0"><mod mode="afsk" bps="1200" f0="1200" f1="2200"
diffenc="1" inlv="8" fec="1" tunelen="32" synclen="32"/><demod
mode="afsk" bps="1200" f0="1200" f1="2200" diffdec="1" inlv="8" fec="3"
mintune="16" minsync="16"/><pkt mode="MKISS" ifname="sm0"
hwaddr="XXXXX-00" ip="44.94.11.8" netmask="255.255.255.0"
broadcast="44.94.11.255" file="/dev/soundmodem0"
unlink="1"/></channel></configuration>
</modem>
```

Replace XXXXX-00 with your own callsign and SSID. But since this will not be used to transmit over the air, I suggest you use N0CALL-12 , sm0 is the interface name, transmit delay (txd) is set at 250 for VOX operation and TX tail to 100.

Now **$ chmod 7555 /etc/ax25/soundmodem.conf** and we are almost done

Edit the /etc/axports so that we have the soundmodule port sm0 in it, by changing it as shown.

$ nano /etc/axport

```
# /etc/ax25/axports
#
# The format of this file is:
#
# name callsign speed paclen window description
#
sm0     PA0ESH-12       1200    255     2       VHF APRS (1200  bps)
```

Note: in the previous document there was an empty line between the last hash (#) and sm0.
Empty lines are not allowed in axports, and yoyr programme will not recognize the port.
In case you wish to use a hardware tnc, add a line like this and make sure the speed between tnc and Raspberry is set correctly (3rd figure 9600 in my case for a TNC-X) – the port name is aprs

```
#
aprs     PA0ESH-12       9600    255     2       VHF APRS (1200  bps)
```

Step 5

Making it all work…..

Testing if soundmodem works…

$ soundmodem

```
ALSA: Using sample rate 9600, sample format 2, significant bits 16, buffer size 4800, period size 144
ALSA: Using sample rate 9600, sample format 2, significant bits 16, buffer size 4800, period size 144
```

If you had the output as above, you're good to go.
When we have sound modem run later on, we will put it's output somewhere were it will not bother us with the verbose output.

The last thing to do is to test if we can transmit.

First we start-up soundmodem
$ soundmodem /etc/ax25/soundmodem.conf -R -M >/dev/null 2>/dev/null&

Then

$ /sbin/route add -net 44.0.0.0 netmask 255.0.0.0 dev sm0

$ /bin/ping -i 10 44.136.8.58 # (ping packets go out on the audio each 10 secs, control-C to stop)

$ **/usr/bin/axcall sm0 pa0esh-10 via pi1twe-2** # (Connect packets go out on the audio)

If you hear the typical aprs chirp, you're good to go to the end of this chapter, and that is to write a small start-up script for the ax25 service.
Now kill soundmodem again
$ killall soundmodem

We'll put the script in somewhere in the path and call it ax25-up.sh

$ **nano /etc/ax25/ax25-up**

enter the following text

```
#!/bin/sh
# Start ax25 with the soundmodem driver using the port sm0
# as defined in /etc/ax25/axports
# Raspberry PI with aprx and soundmodem
# PA0ESH - updated August 6th 2013
echo "Starting soundmodem, please wait...."
soundmodem /etc/ax25/soundmodem.conf -R -M >/dev/null 2>/dev/null&
sleep 5
/sbin/route add -net 44.0.0.0 netmask 255.0.0.0 dev sm0
echo "Soundmodem started at sm0"

# Uncomment the following lines in case of an TNC - check the TNC port by
dmesg | grep tty
# it is assumed that the TNC modem communicates with 9600 baud with the
Raspberry PI
# /usr/sbin/kissattach /dev/ttyUSB0 aprs 44.24.250.250
# /usr/sbin/kissparms  -p aprs  -t 500  -s 200    -r 32      -l 100  -f n
# /sbin/route add -net 44.0.0.0 netmask 255.0.0.0 dev aprs
# echo "ax25 tnc modem started at aprs"
sleep 3
# listen for stations heard
/usr/sbin/mheardd
sleep 1
# to start aprx also automatically - uncomment the following line
#/sbin/aprx -f /etc/aprx.conf restart
# do not forget to adapt the config file with your own data.
```

And to complete the process, create a ax25-down script
$ **nano /etc/ax25/ax25-down**

enter the following text

```
#!/bin/sh
# Stop ax25 with the soundmodem driver using the port sm0
# as defined in /etc/ax25/axports
# Raspberry PI with aprx and soundmodem
# PA0ESH - August 6th 2013
killall soundmodem && echo "Soundmodem closed"
killall kissattach && echo "ax0-9 (kissattach) closed"
# to stop aprx also automatically - uncomment the following line
# /sbin/aprx stop && echo "aprx stopped"
```

Finally we would like a script which handles the autostart of ax25, and aprx for that matter

We call it aprs and put it in /etc/init.d

$ nano /etc/init.d/aprs

```
### BEGIN INIT INFO
# Provides:          aprs
# Required-Start:    $local_fs $network
# Required-Stop:     $local_fs
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: aprs start stop
# Description:       starting and stopping ax25 and aprx
### END INIT INFO# If you want a command to always run, put it here

# Carry out specific functions when asked to by the system
case "$1" in
  start)
    echo "Starting ax25 services"
    /etc/ax25/ax25-up
    sleep 5
    /sbin/aprx start
    ;;
  stop)
    echo "Stopping ax25 servcies and aprx"
    # kill application you want to stop
    killall aprx
    /etc/ax25/ax25-down
    ;;
  *)
    echo "Usage: /etc/init.d/aprs {start|stop}"
    exit 1
    ;;
esac

exit 0
```

To register your script to be run at start-up and shutdown, run the following command:

- **cd /etc/init.d && insserv aprs && update-rc.d aprs defaults**

If you ever want to remove the script from start-up, run the following command:

- update-rc.d -f aprs remove

Step 6

Installation of APRX

See their website at http://wiki.ham.fi/Aprx.en

Download the latest tarball from here
http://ham.zmailer.org/oh2mqk/aprx/
Best is to do that in your /tmp directory

$ cd /tmp
$ wget http://ham.zmailer.org/oh2mqk/aprx/aprx-2.07.svn593.tar.gz

$ tar -xvzf aprx-2.07.svn593.tar.gz
$ cd aprx-2.07.svn593
$ ./configure
$ make clean && make && make install

$ mkdir –p /var/log/aprx

Now read the aprx manual and make your own settings in **/etc/aprx.conf**

You can test aprx by the command

$ aprx –ddv

===== Get the latest aprx from the repository=====
It is possible to compile your own aprx from trunk on the raspberry, enabling you to get the latest version….
In that case follow the next steps:

Instead of downloading the tar ball, do the following:
- Login as root.
- Install subversion
- Then cd into /usr/src
- Get the latest aprx build from the svn repository
- Change into the trunk directory
- Create the SVNVERSION file – empty
- Configufre and make the executables.
- Commands are hereunder……

$ apt-get update && apt-get install subversion
$ cd /usr/src
$ svn co http://repo.ham.fi/svn/aprx/trunk/
$ cd trunk
$ touch SVNVERSION
$ ./configure
$ make clean && make && make install

Then modify the /etc/aprx.conf file and test with aprx –ddv

Finally create the startup file in /etc/init.d/

Erik

See what errors appear, correct them, and you should be up and running.

Adjust TX delay and tail in the soundmodem.conf file

Step 7

Webbased RF/APRSIS log

KISSOZ has made a nifty log-page, here is how I solved it for the Raspberry.

Install lighttpd
$ apt-get install lighttpd txt2html
the website is located at /var/www

$ cd /var/www
$ mkdir –p aprx
$ ln -s /var/log/aprx /var/www/aprxlog
$ cd aprx
$ wget http://www.kissoz.dk/aprx/logserver/genindex.sh
$ wget http://www.kissoz.dk/aprx/logserver/tac.sh

For the Raspberry, you have to adapt the genindex.sh file. Open it with your favourite editor and change the contents as follows: (note that you have to check the path to your webserver – which is in my case /var/www, but that could be something else….

$ nano /var/www/aprx/genindex.sh

```
touch /var/www/aprxlog/aprx-rf.log
tail /var/www/aprxlog/aprx-rf.log -n40 > /var/www/aprxlog/rflog.txt
/var/www/aprx/tac.sh /var/www/aprxlog/rflog.txt > /var/www/aprxlog/rflog1.txt
txt2html -pb 0 --italic_delimiter "" --bold_delimiter "" --underline_delimiter "" --extract
/var/www/aprxlog/rflog1.txt > /var/www/aprxlog/rflog.txt
cat /var/www/aprx/index_head.txt > /var/www/aprxlog/index.html
cat /var/www/aprxlog/rflog.txt >> /var/www/aprxlog/index.html
cat /var/www/aprx/index_end.txt >> /var/www/aprxlog/index.html
```

and save the file….

All the above created a directory in the webserver path and a symbolic link to the location of the log files, the log files disappear when the router is restarted as they reside in a memory disk.
Last imported the script that generates the public webpage. (not very fancy, but efficient)
Next get the script that converts the log content, so it can be displayed on a web page:

$ wget http://vgoenka.tripod.com/unixscripts/text2html.awk.txt
$ mv text2html.awk.txt text2html.sh

$ chmod 755 text2html.sh
$ chmod 755 genindex.sh
$ chmod 755 tac.sh
$ wget http://www.kissoz.dk/aprx/logserver/index_head.txt
$ wget http://www.kissoz.dk/aprx/logserver/index_end.txt
$ wget http://www.kissoz.dk/aprx/logserver/index.html

Change the file tac.sh by changing the first line from

#!/bin/ash
into
#!/bin/bash

Now we are almost ready.... all we have to do is to edit the cron job to include an extra line:
$ crontab -e
The nano editor comes up add the following lines at the end.

*** * * * /var/www/aprx/genindex.sh**
@reboot mkdir –p /var/log/aprx

Ctrl-O saves the file and Ctrl-X exits

Now start it the first time by hand ……

$ mkdir –p /var/log/aprx && /var/www/aprx/genindex.sh

and then auto:

$ /etc/init.d/cron restart

Look at localhost/aprx for the results.

Happy iGating & Digipeating

Erik, pa0esh at amsat.org