# DIXPRS ON A RASPBERRY

05/03/2014 10:48:45 pm

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.
Since beginning of December there is a possibility to add PTT. See the article

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
- Dantracker N7NIX
- 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 && sudo wget
http://www.pa0esh.nl/aprs/rpi/aprx.sh && sudo chmod 755 aprx.sh && sudo ./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 links 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 bot 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**

Since we are going to install dixprs, we require python 2…. (already installed, but also the python-serial as well as sqlite3 modules

**$ sudo apt-get install python-serial sqlite3  libax25 build-essential python-dev libax25-dev**

Since you most likely wish to run the digipeater autonomous, also in case of a restart, I suggest we move to the root user right now

**$ sudo su**

On LINUX DIXPRS can communicate to a KISS TNC directly via a serial interface. However it is advised to use TNC's and modems via AX.25 stack. It makes other type modems, like USCC card, BAYCOM modem, etc. usable with DIXPRS. It requires an additional Python module installed, developed for DIXPRS, called pyax25.
Note: procedure requires libax25, Python development files and basic development tool chain with GCC installed.  They were installed above (libax25 & python-dev)

Download pyax25.tar.gz and extract to a temporary folder.

- mkdir pyax25
- cd pyax25
- wget https://sites.google.com/site/dixprs/downloads/pyax25.tar.gz
- tar -xzvf pyax25.tar.gz

Then enter

./compile_ax25.py build
./compile_ax25.py install

Check installation as follows:

Start Python (simply type python) and type

import _ax25

If everything is OK, only prompt displayed, no error message.

For more dixprs information see the website from Béla Márkus, HA5DI,
The most informative Google groups about dixprs is located at
https://groups.google.com/forum/?fromgroups - !forum/dixprs

## Step 3

Create the dixprs directory and cd into it
**$ mkdir –p /usr/local/dixprs && cd /usr/local/dixprs**
Download the source code from using wget.....
**$ wget https://sites.google.com/site/dixprs/downloads/dixprs-2.2.2.tar.gz**
or from the download page on the website
Unpack the tar ball
**$**

Now chmod all files
$ chmod 755 *

## Step 4

Soundmodem and soundmodem tools installation
**Skip this paragraph if you are using AX25 or KISS with TNC modem !**

$ **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...

Here is the link if it still exists.

Stick the USB dongle into the Raspberry and check what you have got.

**$ 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 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

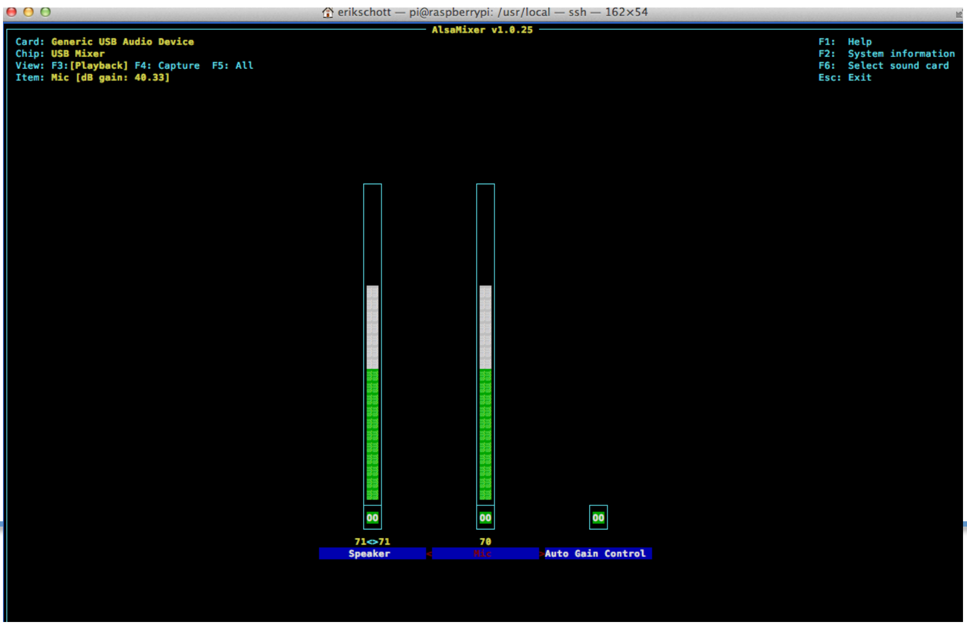
```
options snd slots=snd_bcm2835,snd_usb_audio
options snd_usb_audio index=0
options snd_bcm2835 index=2
```

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
```



```
erikschott — pi@raspberrypi: /usr/local — ssh — 162×54
AlsaMixer v1.0.25
Card: Generic USB Audio Device                                    F1:  Help
Chip: USB Mixer                                                   F2:  System information
View: F3:[Playback] F4: Capture  F5: All                          F6:  Select sound card
Item: Mic [dB gain: 40.33]                                        Esc: Exit

        00              00              00
      71<>71            70
     Speaker    <      Mic      >Auto Gain Control
```

Now we check the sound system:

Start 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

Replace it with the following content :

```
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 5

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 directory /etc/ax25.

**$ nano /etc/ax24/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. I suggest you use N0CALL-12 where N0CALL is your callsign.

- sm0 is the interface name, transmit delay (txd) is set at 250 for VOX operation and TX tail to 100.

Now **$ chmod 7555 /etc/ax24/soundmodem.conf** and we are almost done

Edit the /etc/axports so that we have the sound module 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)
#2      OH2BNS-9        38400   255     7       TNOS/Linux  (38400 bps)
```

## Step 6

Making it all work.....

Testing if soundmodem works… Enter the following command:

**$ 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 command

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&**

Note: this method seems to give some problems lately – another option is to use a screen:

**$ screen –dmS sm soundmodem**

Then

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

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

3. $ **/usr/bin/axcall sm0 k7xyz via sea** # (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** or **$ killall screen**

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 dixprs and soundmodem
# PA0ESH - April 20
#soundmodem /etc/ax25/soundmodem.conf -R -M >/dev/null 2>/dev/null&
screen –dmS sm soundmodem
sleep 5
/sbin/route add -net 44.94.11.0 netmask 255.255.255.0 dev sm0
echo "Soundmodem started at sm0"
# to start dixprs also automatically - uncomment the following line
# screen –dmS dixprs usr/local/dixprs -c /usr/local/config.txt
# 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 dixprs and soundmodem
# PA0ESH - April 2013
killall soundmodem
#killall screen
# to stop dixprs also automatically - uncomment the following line
# killall screen
```

## Step 7

Now it's time to adjust the dixprs config file for use with ax25
It is located at /usr/local/dixprs and we use the config-ax25.txt sample.
Either rename it to config.txt or call this config file with startup of dixprs.

After adapting your personal data, such as call sign etc, pay attention to the radio part. It should look ow as follows:

```
[RADIO]
# Mandatory parameters
# Interface type
INTERFACE=AX25
# Device; as listed by ifconfig; it is nbot the ax.25 port name!
DEVICE=sm0
```

If you did not use the soundmodem , the edit the KISS config sample file and continue from there. Make sure you have your settings for the serial port correct, as well as your call+ssid, lat and lon etc etc.

Rename the configuration file to **config.txt**

DIXPRS is supposed to run in its own screen, but before that we test it

```
$ /usr/local/dixprs/dixprs.py –c /usr/local/dixprs/config.txt
```

If all is well, you should see something like this….

```
*** DIXPRS 2.2.2 - 05-09-2012 - (pid=8473)

*** AIR0  process started [TNC-X FTDI USB to Serial port] (pid=8476))
*** IGATE process started [host=euro.aprs2.net, port=14580] (pid=8477)
*** WEBSR process started [port=9999] (pid=8478)
*** Connection established

21:10:55z AIR0       <- LOCL    PA0ESH-2>APDI22,WIDE1-1,WIDE2-2:>DIXPRS 2.2.2 up and running
*** Connection successful to T2GYOR
21:10:59z AIR0       <- LOCL    PA0ESH-2>APDI22,WIDE1-1,WIDE2-2:>Connected to T2GYOR count 1/0
```

## Step 8

Normally you would want your Raspberry to start dixprs automatically at boot.

The next little script will take of it, but before that you have to install screen.

**apt-get install screen**

Cd into /etc/init.d
**$ cd /etc/init.d**
Then create the dixprs start-up file
**$ nano dixprs**
Copy the content below into it, and write it to file.

```
#! /bin/sh
# /etc/init.d/dixprs

### BEGIN INIT INFO
# Provides:          dixprs
# Required-Start:    $remote_fs $syslog
# Required-Stop:     $remote_fs $syslog
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Simple script to start dixprs at boot
# Description:       A simple script from www.stuffaboutcode.com which will
start / stop a program a boot / shutdown.
### END INIT INFO

# If you want a command to always run, put it here
cd /usr/local/dixprs

# Carry out specific functions when asked to by the system
case "$1" in
        start)
                echo "Starting dixprs in a screen"
                # run application you want to start
                screen -dmS dixprs /usr/local/dixprs/dixprs.py -c
/usr/local/dixprs/config.txt
                ;;
   stop)
                echo "Stopping dixprs screen"
                killall screen
        ;;
        *)
   echo "Usage: /etc/init.d/dixprs {start|stop}"
   exit 1
   ;;
esac
```

exit 0

Then
**$ chmod 755 dixprs**

Test if the script works….

**$ ./dixprs start**

**screen –r dixprs**, should show you the previous start-up information.

```
*** DIXPRS 2.2.2 - 05-09-2012 - (pid=8473)

*** AIR0  process started [TNC-X FTDI USB to Serial port] (pid=8476))
*** IGATE process started [host=euro.aprs2.net, port=14580] (pid=8477)
*** WEBSR process started [port=9999] (pid=8478)
*** Connection established

21:10:55z AIR0      <- LOCL    PA0ESH-2>APDI22,WIDE1-1,WIDE2-2:>DIXPRS 2.2.2 up and running
*** Connection successful to T2GYOR
21:10:59z AIR0      <- LOCL    PA0ESH-2>APDI22,WIDE1-1,WIDE2-2:>Connected to T2GYOR count 1/0
```

Leave the screen by Ctrl-a-d (then dixprs will be running).
Kill it by
$ **./dixprs stop**
but from outside the screen with the name dixprs…..

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

**$ update-rc.d dixprs defaults**

Note - The header at the start is to make the script LSB compliant and provides details about the start up script and you should only need to change the name. If you want to know more about creating LSB scripts for managing services, see http://wiki.debian.org/LSBInitScripts

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

**$ update-rc.d -f  dixprs remove**

## Bash file

Loginto your Raspberry as pi. then execute sudo su

```
apt-get update
sudo apt-get install python-serial sqlite3 libax25 build-essential python-dev ax25-tools
libax25-dev
mkdir pyax25
cd pyax25
wget https://sites.google.com/site/dixprs/downloads/pyax25.tar.gz
tar -xzvf pyax25.tar.gz
./compile_ax25.py build
apt-get install python-dev
./compile_ax25.py install
mkdir –p /usr/local/dixprs && cd /usr/local/dixprs
wget https://sites.google.com/site/dixprs/downloads/dixprs-2.2.2.tar.gz
tar -xvzf dixprs-2.2.2.tar.gz
chmod 755 *
```