Repeater Linking Project.

Official User Manual.

Image Version 3

Revision 2.40, Printed 27/07/2019

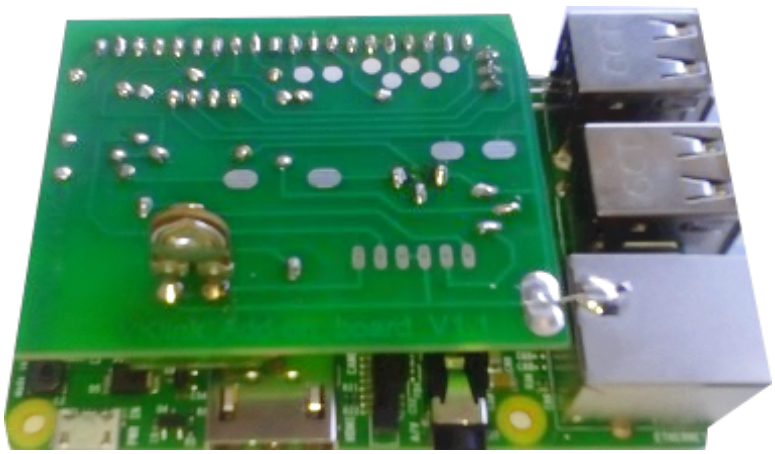


Table of Contents

Table of Contents

About the project

The Idea

For quite a number of years, VK3VS has been investigating software based repeater controllers. It started off using a PicAxe 08m, then went onto a Picaxe 18M2. These worked great with 20mA current draw. The 08m controller gave a simple control for PTT, RX indicator, a subtone enable, a PTT for link radio and a CW ID. The 18M2 gave all this, plus extra pins for fans and battery voltage monitoring and so forth. While These were fun and worked, they were still very limited.

Then along came the Raspberry Pi - Great, a embedded PC that draws around 80mA at 12V. Perfect. Or so I thought.

Initial Attempts

Version 0.1: First of all there was an attempt using the Gpio's and a sound dongle running inside a bash script. This proved to not be very efficient as the script was limited in its ability to make intelligent decisions. And generating more was a pain.

Version 0.2: The second attempt came to me after I had set up a PBX using asterisk. reading some old documentation, I found a program called app_rpt that run under asterisk 1.6, It took several goes to compile the software under Pidora, as asterisk was designed for red-hat based systems. I got it running, however the audio was very broken and there was no way to control the radio ptt as there was no parallel port.

Version 0.3: I found Jim Dixons variant of app_rpt as a fork from the original as asterisk had decided to not include it in future releases, he had forked it an added the chan_usbradio driver which used gpio's on the sound fob for the control. This I thought would work, so again I tried compiling it on Pidora. Well it worked, but again audio was broken but I could control the radio!

Version 0.4: After reading more and more about app_rpt and chan usb, the common consensus was the Raspberry Pi did not have the horsepower to do DSP. so I went looking at the Simple USB channel for a way of getting the radio to work without broken audio. See one of the problems is, the later kernels that run do not use OSS, instead they use alsa. so I thought the broken audio problem was the Alsa-OSS emulation, even though it was at Kernel level. I compiled simple usb and damn. still broken audio.

Version 0.5: I decided to purchase an Odroid U3, to attempt to compile it. After months of mucking around to get kernel source, as I needed to compile Zaptel and incorporate the oss emulation into the kernel, I made a head road. the software compiled, with some ancient hacks to deal with udev and a piece of software that did not know udev existed, it ran, and clean audio. Now the Odroid takes a dump on the Pi (and even the Pi2) on specs, but is cheap. however a little power hungry. Thought this is better, and possibly the way forward.

Version 0.6: Not content with a couple of Pi's sitting there not doing what I wanted them to do, I had another go... This time, I thought, I wonder if I make a custom kernel, will the thing work. So, I decided to use raspbian, as it is written aimed at the Pi. I stripped down the kernel, removing anything that I did not need and stuff I thought would make a pain in the neck out of itself. In particular power scaling. Then I run into another problem. The kernel has had some more ancient stuff removed from it, which means zaptel would not work ever.... so I had to think dahdi (same thing different name), but lucky for me, the source of app_rpt had this already built in to replace zaptel. 23 hours of compiling later, it was born. My Repeater was born on the Pi. the usb audio was a little broken, but the simple audio was flawless.

Linking: After it worked, the idea came to me, well now it works, why not use the linking capabilities as well. And there you have it, the VK repeater link was born. Version 1 was released not far after this idea, which had a heap of scripts to co-ordinate connections. The first nodes being VK3VS to VK7DB, nodes 100 and 101 respectively.

After that came a server to co-ordinate it, assigning of node numbers to states, fixing broken code and adding useful bits to it.

Initial Thoughts: There has been quite a lot of interest sparked by playing with this stuff, but as always with any projects, there is some that are quite loud in their jealousy. To quote a glorified CB technicians comment via email:

" Most of Matts ideas are crap and the rest are mediocre at best. "

Think I'll put this little project in the mediocre category.

How it works

Startup

When you power up your node, it fires up and looks for an active internet connection.

If it cannot find one, it refuses to start. This is changeable for offline nodes (ie Remote Nodes).

When it has found an active internet connection, it downloads scripts to allow for ease of change of settings.

Co-ordination

When the program starts, it reports its status to the VKLink server http://status.vklink.com.au website.

If the node has been enabled by admin, it is put into the database that other nodes look at to find connections.

It also reports its current status, ie, online, whether someone is txing into it, and various other bits of info.

Security

The security is derived from IP addresses and parts of the script that is downloaded. If the reported IP and the found IP is different, it is kicked off.

If you have a dynamic IP address, a DNS name has to be assigned to it so a reverse lookup can be done to verify the IP. A script has to be manually added for this.

Connections

If all the above is correct, you type a Connection Command and a Node Number from the http://status.vklink.com.au , your node looks at the node list generated by the VKLink server and attempts to connect to it.

All the control/audio packets are sent via the one UDP port, 4569, meaning a great security increase for your personal network.

Conferences

No such thing. You simple connect to nodes you want to talk to. Look at the map first and see what is already connected to what, and connect to the closest node.

Version Information

Current Version

The Current version of the VKLink image is Version 3.0. This image is for the Pi 2 and Pi 3 and Pi 3B+.

Major re-write with:

- Updated Kernel to use with the new version of the Raspi 2, 3 and 3B+.
- Iptables used as a firewall
- Raspbian Stretch as the base distro
- Everything is controlled via systemd (systemctl)
- Apache as the webserver

To get the image, go to http://www.vklink.com.au/download.php

Previous Versions

v2.0 Kernel re-write to suit the new version 2 and 3 of the raspi. Allmon and web terminal as standard, Read-only filesystem, slightly improved device drivers.

v1.4 Major Changes to this version, a major re-write of the rpt_app to parse node lists better, added direct functionality with the status website, and allowed another channel driver. This is the last version for the Pi 1.

V1.3 vk_repeater_network_v1-3(-pi2).zip - moved a fair bit of day to day stuff to a ramdrive to save the SD Card

V1.2 vk_repeater_network_v1-2.zip - changed scripts for smoother starting

V1.1 vk_repeater_network_v1-1.zip - initial version

Upcoming Versions

No new upcoming versions at the moment unless the Raspberrry Pi foundation makes changes to their hardware again that we have to catch up upon.

Required Parts

Parts list – using cm108/119 GPIO’s

Simple parts are required to make the controller work:

- Raspberry Pi B, B+, B2 or Pi 3
- Power supply with micro USB connector (either from 240V or 12V depending on the application.
- Sound Fob. These have to be a GENUINE CM108, 109 or 119 Fob if you want to use the fob GPIO’s. Cheap Chinese fobs will work if you use the Raspberry Pi Gpio’s
- The following discrete components:
    - 1 1n914 Diode
    - 1 10k resistor
    - 1 NPN transistor (value non critical, but a BC547 suffices)
    - 1 500k Trimpot.

To use the genuine cm108/119 and its gpios, skip down to Modifying the Fob. The current Channel drivers allow the Raspberry Pi Gpio’s to be used..

Updated Parts List – Using Raspi GPIO’s

As this is a rolling project, things are bound to change. We are proud to announce that the chan_simpleusb and chan_usbradio drivers have been modified to work in conjunction with the Raspberry pi and cheap chinese sound cards.

- Power supply
- Any of the usb sound fobs available on the internet.
- The following discrete components:
    - 1 1n914 Diode
    - 1 10k resistor
    - 1 NPN transistor (value non critical, but a BC547 suffices)
    - 1 500k Trimpot.
    - 1 100k resistor

To see how the Raspberry Pi is wired using cheap chinese dongles see Raspberry Pi GPIO wiring

Modifying the Fob

To get audio into and out of the node, plus get some signals of interest to and from the radio we need to modify the sound fob.Only a GENUINE CM108, 109 or 119 IC will work. The cheap chinese crap will not work.From left to right, a pair of real CM119 Audio fobs from syba.

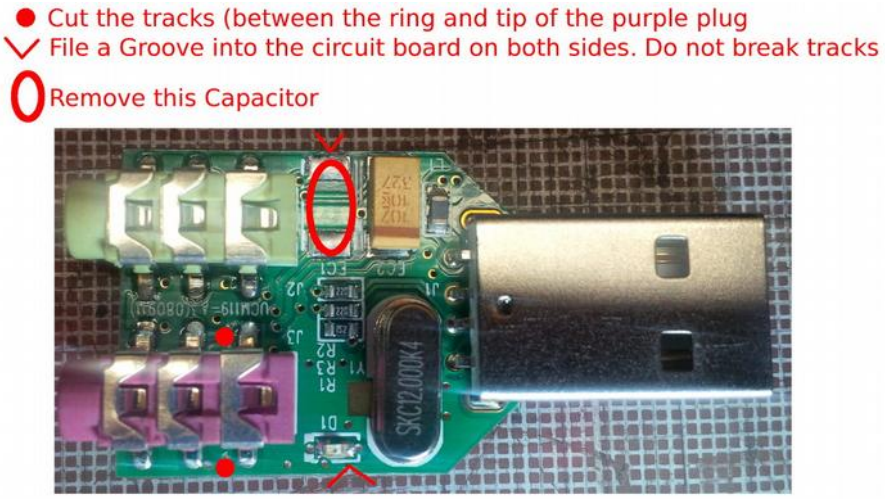
The middle two, even though they report as CM108, are not CM108, just cheap copy crap. Even if there is gpio pins, you cannot get to them. Do not buy these, even though they are cheap at even $1.00each free postage.

The last one, even though sold as a 7 channel surround sound CM108 fob from dxzone is crap and does not even report as a CM108.

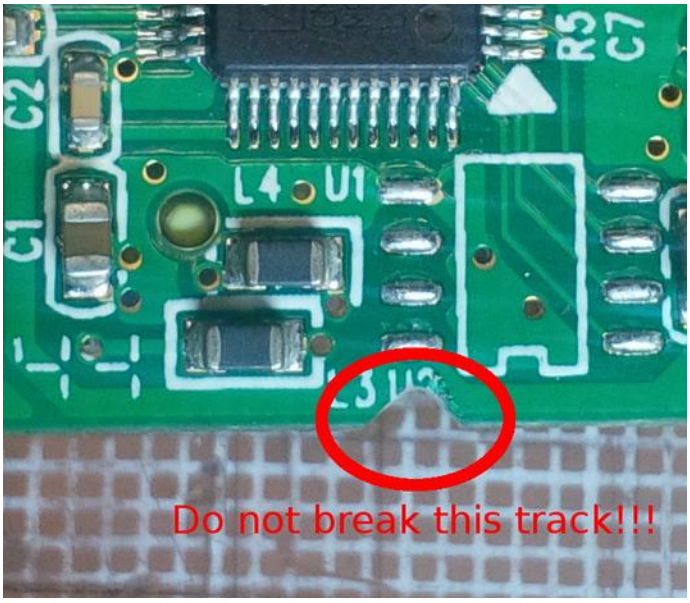
BUY A DECENT FOB if you want use the fob GPIO's

Onto building it. Crack open the case (slight pressure in a vise will do it).

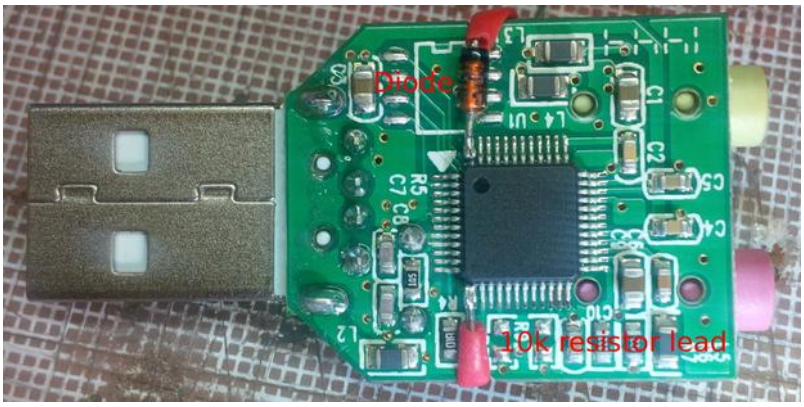
Two tracks need cutting, 1 capacitor will need removing and a couple of grooves will need to be cut into the circuit board to allow wiring from the other side.



Be careful not to break one track while cutting the groves, the rest it does not matter as they are linked earths anyway.



The trickiest part of the build. you need to solder (with a piece of sheathing over the lead) to the pins on the IC as pictured. Then bend the leads with sheathing through the groves. The reason we need sheathing is because of the exposed earth cuts in the board.



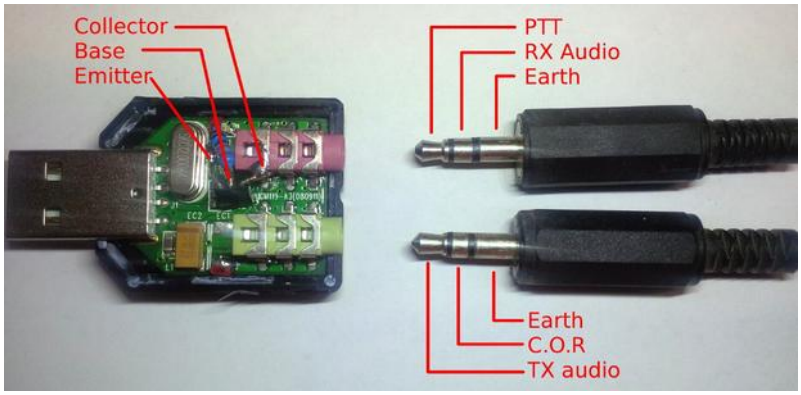
We cut the groves so the unit will fit back into its case.

Clip the unit back into its case so the pressure of the case stops you from ripping the leads off the IC or worse...

Solder the free leg of the diode to the pad closest to the side of the board where the capacitor was removed.

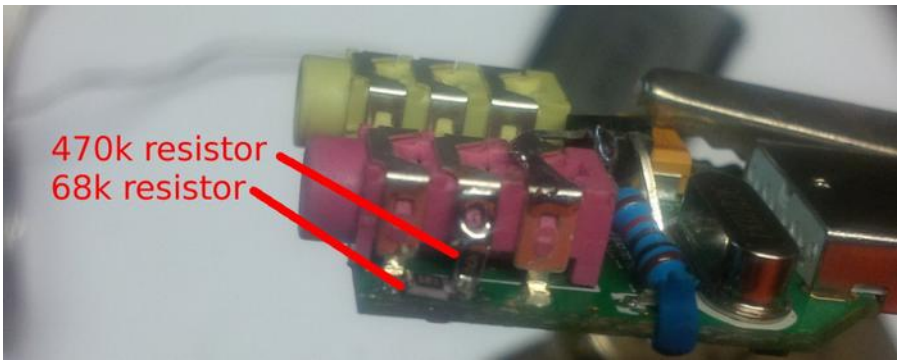
Get the NPN transistor, solder the collector to the tip of the purple socket, solder the base lead to the free end of the resistor, and solder the emitter to the square earth pad beside the crystal. push the whole plot down below the height of the sockets so the case clips together again.

The 500k pot needs to be put in the audio line from the receiver to the Mic socket, this can be done inline or in the radio itself. what we are trying to achieve here is a high impedance input with a voltage divider for the levels on the mic socket. We are amateurs a picture should not need to be drawn of this. One side of the pot to the receivers audio, the other side to earth and the centre pin to the audio fob.



This should give us a neat and tidy audio interface between radio and Pi.

For those feeling brave, there is a voltage divider that has been used on most radios with great success, using the software to modify the levels. Lift the leg of the ring of the pink connector, closest to the side of the board. cut the tag off the bottom and bend whats left under the connector body for security. slide a little knife in underneath and cut the track between the two sides. Now solder a 470k smd resistor between the pad and where the leg has been folded. then install a 68k resistor between the pad and earth..... Good luck. it is doable though



Installing the Image

Getting the Image

Go to http://www.vklink.com.au/download.php , fill out your details and download the image.

There are four Image files:

- A version for the Raspberry Pi B or B+, and
- Version 2 for the Raspberry Pi B2 and Pi 3.
- Version 3-Pistar – This is an experimental hybrid vklink/pistar image
- Version 3 for the Pi 2, 3 and 3B+. This manual is for this image.

Burning the Image

Once you have downloaded the image, it has to be burnt to the SD card. This means, the image is copied to the card, byte for byte. Copying the .img file to the SD card with windows explorer will render you with a DOS formatted Card with one big file on it.

Burning in Winblows

I cannot test this, as I do not have access to a Winblows Machine, but apparently Winblows 8 will not work.

You need to download a program called win32diskimager. The genuine program is available from sourceforge.

A copy of that program is stored here: http://www.vklink.com.au/manual/Win32DiskImager-0.9.5-binary.zip .

Unzip the contents of the file, and run the .exe file within. Ignore any carry-on from Winblows that the program is untrusted etc, etc as most open source programmers do not have the funds to go through all that crap. Not only that, putting the code into such a program will bloat it.

1. Load the Program
2. In the Left side of the window, select your .img file.
3. In the right side of the window, select your SD card drive.
4. Press Write.

5. Walk away and make a coffee or something.

When its finished, insert it into the Pi, plug a monitor and keyboard into the pi and apply power

Burning in Linux (And Mac I think)

Insert an SD card of at least 2GB into your computer. Run:

```
df -h
```

to find your SD drive. for this example, the SD drive is going to come up as /dev/mmblk0

Unzip the image.

Run:

```
dd if=vklink-v3-0.img of=/dev/mmblk0 BS=1M
```

After a period of what seems forever the prompt will come back

Run:

```
sync
```

to flush the cache, and now insert the image into your Raspberry Pi.

Initial Setup

Booting

Now that you have your image in your pi, it is time to turn the power on.

You have to make sure your pi is plugged into a network cable with internet access or the Repeater Application will refuse to start

While the Pi is booting, there will be quite a bit of rubbish on the screen. Take no notice of any of it.

There are warnings as things have been turned off that either waste CPU or are not needed.

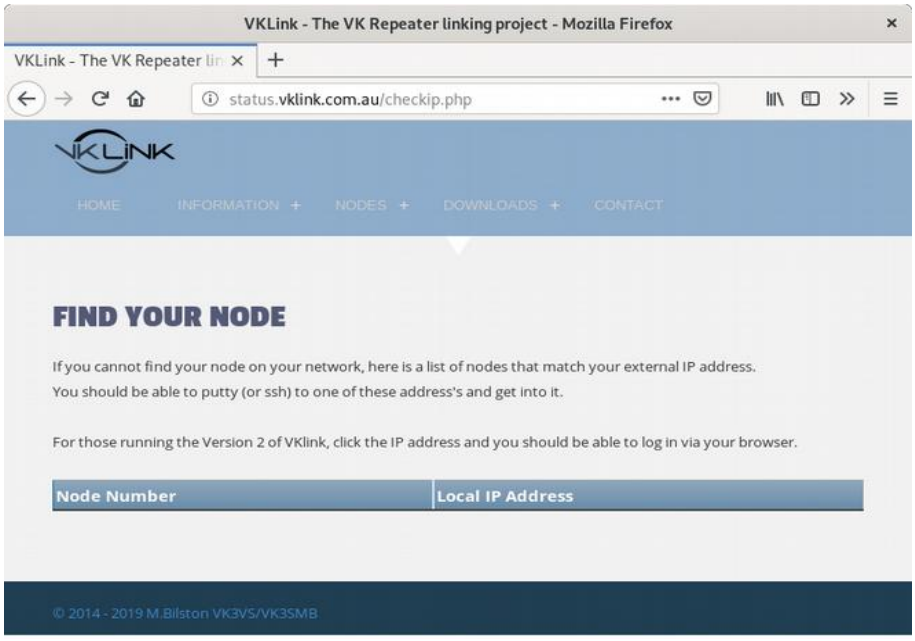
You also need your dongle plugged in so it can see it.

Web interface

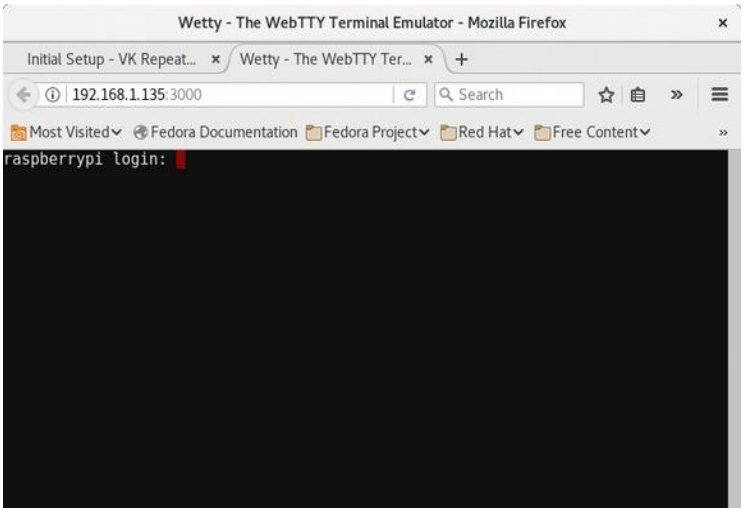
From VKLink version 2, you can simply use a web browser to configure the node.

After your node has booted up, within 10 minutes it will report its internal IP address to our server.

You go to http://vklink.com.au/checkip.php and it should come up with all of your nodes registered to your IP address.



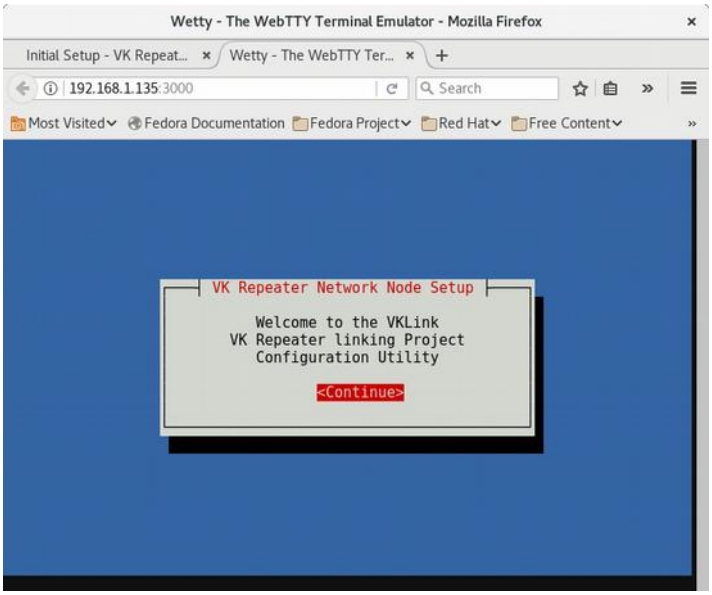
Simply click on the internal IP address and you will go to a browser window where you can log in.



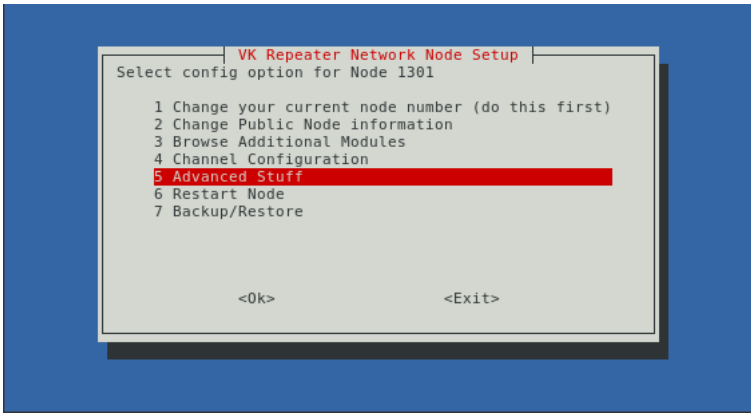
To log in to the Node,

- Username is pi, and
- Password is raspberry

Once this is entered, after a short pause, the VKSetup Main Menu will load:



Press {enter} and you will be presented with a menu as follows:

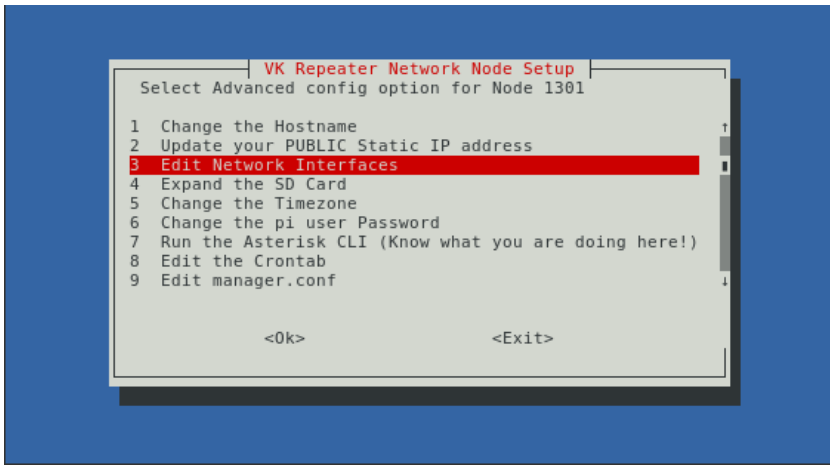


From here on you need to use the arrow keys on your keyboard (remember those!) to navigate around.

Network Configuration

In order to keep your node at the one IP address, you need to manually edit the network file to give it a static IP address. Google will help here. An example will follow, however it will not suit all networks.

In the setup menu, scroll down to *Advanced Stuff*, press {enter}



Scroll down to *Edit Network Interfaces* and press {enter}.

You will now have a file open in nano similar to this:

```
auto lo

iface lo inet loopback
iface eth0 inet dhcp

auto eth0
#iface eth0 inet static
#your static IP
#address 192.168.1.12
#your gateway IP
#gateway 192.168.1.1
#netmask 255.255.254.0

allow-hotplug wlan0
auto wlan0
iface wlan0 inet manual
wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf
iface default inet dhcp
```

What needs to be done here, is the

```
iface eth0 inet dhcp
```

needs to be commented out (put a # in front of it), and the static stuff needs to be uncommented and changed to your network.

The easiest way to do this is go to another computer and type *ipconfig* in a windows command prompt, or *ifconfig* in a Linux terminal.

At the very least we need an address, gateway and netmask. Set these, Press, {Ctrl}-X, y, {enter} to save the changes. It now look something like this:

```
auto lo

iface lo inet loopback
#iface eth0 inet dhcp

auto eth0
iface eth0 inet static
#your static IP
address 192.168.1.12
#your gateway IP
gateway 192.168.1.1
netmask 255.255.254.0

allow-hotplug wlan0
auto wlan0
iface wlan0 inet manual
wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf
iface default inet dhcp
```

You will now be back in the menu.

Timezone

Scroll down to Change the Timezone, Press {enter}.

Follow the prompts to change the timezone to the correct timezone. Either UTC or Australian time.

And Finally,

Expanding the SD Card

Scroll down to Expand the SD Card, and Press {enter}

Follow the prompts. Then Exit out of the menu system and it will ask you to reboot. Allow this to happen.

At this point you can also change the default password. Write it down.

Requesting a Node Number

Now is the time to request a node number. This can be done by the following:

- log into the forum http://forum.vklink.com.au/viewforum.php?id=31 and request a node number.
- Joining the facebook group https://www.facebook.com/groups/vklink/ , or
- send an email to matt@vklink.com.au

There is still more things to do while waiting for your node number.

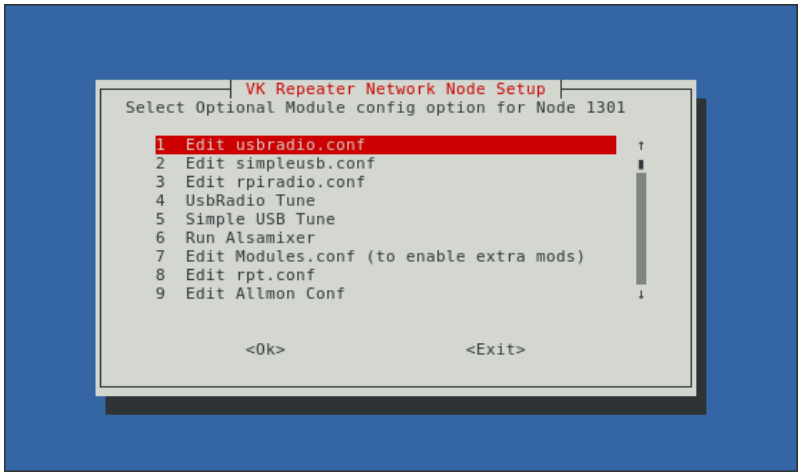
Choosing a Channel Driver

There are 3 Channel drivers available for the Raspberry Pi, the first two have been well and truly tested.

The third is still under development and requires some major work before I am happy with it. It connects and does everything it should do, however there is some funny stuff happening with the audio and it drops the PTT too quickly on the other end. But it is usable if someone wants to play.

Editing the Config files

Every Config file needed to be edited is able to be done via the Main Setup Menu.



Whenever a change is done to a channel driver file the application should be restarted. The Main Setup Menu does this automatically.

Chan_SimpleUSB

This is the driver that works every time, for every version of the Pi. It has basic RX and TX signalling, and a limited amount of audio processing. The configuration file has all the information on setting the channels, with comments explaining it all.

The important things here to note is rxboost=, carrierfrom= and the correct configuration for the Raspberry Pi.

```
;
; SimpleUSB configuration
;
[general]

[usb]
piversion2=1          ; set to 1 for a Raspi 2 or 3, 0 for all others
israspi=1             ; set to 1 when its a pi, leave at 0 for others
pttpin=4              ; Transmitter pin
corpin=2              ; Carrier RX pin
;ctcsspin=3           ; CTCSS RX pin (in this example it is not used)

eeprom=0

hdwtype=0             ; Leave this set to 0 for USB sound fobs modified using
                      ; the instructions from the VKLink manual.

rxboost=1             ; 0 = 20db attenuator inserted, 1= 20db attenuator removed
                      ; Set to 1 for additonal gain if using a low-level receiver output

carrierfrom=usb       ; no,usb,usbinvert,rpi,rpiinvert
                      ; no - no carrier detection at all
                      ; usb - from the COR line on the modified USB sound fob
                      ; usbinvert - from the inverted COR line on the modified USB sound fob
```

```
                    ; rpi – from the corpin on the raspi
                    ; rpiinvert – from the inverted corpin on the raspi

ctcssfrom=no        ; no,usb,usbinvert,rpi,rpiinvert
                    ; no - CTCSS decoding, system will be carrier squelch
                    ; usb - CTCSS decoding using input from USB FOB
                    ; usbinvert - from the inverted CTCSS line on the modified USB sound fob
                    ; rpi - from the corpin on the raspi
                    ; rpiinvert – from the inverted corpin on the raspi

txmixa=voice        ; Left channel output: no,voice
                    ; no - Do not output anything
                    ; voice - output voice only

txmixb=no           ; Right channel output: no,voice,tone,composite, auxvoice
                    ; See txmixa above.

invertptt=0         ; Invert PTT 0 = ground to transmit, 1 = open to transmit
                    ; This is the collector lead of the 2n4401 on the modified
                    ; usb sound fob.
                    ; please refer to the howto for the procedure to do this.

duplex=1            ; ***DO NOT TOUCH***   Full Duplex

plfilter=0          ; Used in conjunction to block PL tones when used in deemphasis mode

deemphasis=0        ; When using flat audio from a receiver

preemphasis=0       ; When wanting flat audio into a transmitter.

rxondelay=0;        ; Uncomment and/or adjust for simplex nodes to eliminate "Ping Ponging"
                    ; or "Relay Racing" (or Bloody Button pushers).
                    ; A positive value here will instruct the usbradio driver to ignore the
                    ; COR line for a specified number of 20mSec intervals following the
                    ; release of PTT. Use this only on simplex nodes, and leave commented
                    ; out for repeaters or other full duplex nodes.
```

The rxboost needs to be changed from 0 to 1 if you cannot get enough volume out of the receiver.

The carrierfrom needs to be changed depending on whether your receiver pulls its RX signal to high or low on receive.

For example the yaesu FT8800 goes high on RX so it would be carrierfrom=usb or carrierfrom=rpi.

To enable the use of simpleusb, the rpt.conf file needs changing to reflect this:

```
rxchannel = simpleusb/usb
```

Where usb is the device defined in the configuration file (in the square brackets).

Chan_USBRadio

Chan_USBRadio is the higher horsepower driver. It can do DSP receiving and generate and decode subtones.

Its configuration file also has annotations for configuring. By using DSP as the receive indicator, it can pull signals below the noise floor.

```
;*************************************************************
;Sample USBRadio Config file

[general]


[usb]
piversion2=1                ; set to 1 for a Raspi 2 or 3, 0 for all others
israspi=1                   ; set to 1 when its a pi, leave at 0 for others
pttpin=4                    ; Transmitter pin
corpin=2                    ; Carrier RX pin
;ctcsspin=3                 ; CTCSS RX pin (in this example it is not used)
```

```
hdwtype=0                    ; leave as 0 for USB sound fobs modified using
                             ; the instructions on the VKlink Manual

rxboost=1                    ; 1 for 20dB boost on the RX stream

txboost=0                    ; 1 for 20dB boost on the TX stream

rxctcssrelax=1               ; leave as 1

rxctcssfreqs=123.0,91.5      ; CTCSS freqs the RX will open with

txctcssfreqs=123.0,91.5      ; CTCSS freqs TX will transmit, it will follow RX tx freqs

txctcssdefault=123.0         ; Default TX CTCSS, the tails and linking will have this freq

rxctcssoverride=0            ; Set to 1 to start in carrier sq mode.

carrierfrom=dsp              ; Where the node gets its received signal from options are:
                             ; no,usb,usbinvert,dsp,vox,rpi,rpiinvert
                             ; no - no carrier detection at all
                             ; usb - from the COR line on the modified USB sound fob
                             ; usbinvert - from the inverted COR line on the modified USB sound fob
                             ; dsp - from RX noise using dsp techniques
                             ; vox - voice activated from RX audio
                             ; rpi – from the corpin on the raspi
                             ; rpiinvert – from the inverted corpin on the raspi

ctcssfrom=dsp                ; where the node gets its ctcss from option are:
                             ; no,dsp,rpi,rpiinvert
                             ; no - CTCSS decoding, system will be carrier squelch
                             ; dsp - CTCSS decoding using RX audio in DSP.
                             ; rxdemod option must be set to flat for this to work.
                             ; rpi – from the corpin on the raspi
                             ; rpiinvert – from the inverted corpin on the raspi

rxdemod=flat                 ; input type from radio: no,speaker,flat
                             ; no - RX audio input not used
                             ; flat - Use RX audio from discriminator (before de-emphasis)
                             ; speaker - use de-emphasized audio

txlimonly=yes                ; Audio limiting with no pre-emphasis on output channel: no,yes
                             ; no - Audio is not limited.
                             ; yes - Audio is  limited.
                             ; Suitable for transmitters with no limiting but with pre-emphasis.

txprelim=yes                 ; Audio processing on left output channel: no,yes
                             ; no - Audio is not pre-emphasized and limited.
                             ; Suitable for use on a microphone input
                             ; yes - Audio is pre-emphasized and limited.
                             ; Suitable for direct connection to an FM modulator

txtoctype=notone             ; Transmit tone control type: no,phase,notone
                             ; no - CTCSS tone encoding with no hang time
                             ; phase - encode CTCSS and reverse phase
                             ; AKA ("reverse burst") before unkeying TX
                             ; notone - encode CTCSS and stop sending tone before unkeying TX
                             ; AKA ("chicken burst")

txmixa=composite             ; Left channel output: no,voice,tone,composite,auxvoice
                             ; no - Do not output anything
                             ; voice - output voice only
                             ; tone - CTCSS tone only
                             ; composite - voice and tone
                             ; auxvoice - auxiliary voice output at headphone level for monitoring

txmixb=no                    ; as above for right channel (not used on our fob)

invertptt=0                  ; Invert PTT 0 = ground to transmit, 1 = open to transmit
                             ; This is the collector lead of the 2n4401 on the modified
                             ; usb sound fob.
                             ; please refer to the howto for the procedure to do this.


rxondelay=0;                 ; Uncomment and/or adjust for simplex nodes to eliminate "Ping Ponging"
                             ; or "Relay Racing" (or Bloody Button pushers).
                             ; A positive value here will instruct the usbradio driver to ignore the
                             ; COR line for a specified number of 20mSec intervals following the
                             ; release of PTT. Use this only on simplex nodes, and leave commented
                             ; out for repeaters or other full duplex nodes.
```

```
jbenable=no

jbmaxsize=200

jbresyncthreshold=1000

jbimpl=fixed

jblog=no

duplex=1
```

Again, the most important items are the rxboost=, txboost=, and carrierfrom= items.

You enable this driver in rpt.conf by setting

```
rxchannel=radio/usb
```

Where usb is the device in the config file.

With both Simpleusb and usbradio drivers, it is important to note that if you set the option *israspi=1*, the driver ignores the FOB gpio's and looks for changes on the raspi gpio's.

Chan_RpiRadio

This is a completely experimental driver written by VK3VS to use Alsa (instead of OSS), and the GPIO pins on the Raspberry Pi.

'It is in Alpha Stage and requires work and as such, it is not included in the images, but it is downloadable via the Main Setup Menu.

Again the conf file has information on what the settings do. The MOST important setting here is piversion2=

You need to set this to 1 for a Pi B2, and 0 for all others. It can make the Pi unusable if this is wrong as the GPIO's are accessed on Ground floor. There is no pretty software protecting you from them.

The config file has all the options explained and is pretty self explanatory.

```
[rpi0]

;Sound Card Crap

input_device=plughw:0,0              ; ALSA input channel
output_device=plughw:0,0     ; ALSA output channel
;silencesuppression=0        ; Silence Suppression

;Raspberry Pi Stuff

pttpin=27                    ; Transmitter pin
corpin=22                    ; Carrier RX pin
ctcsspin=17                  ; CTCSS RX pin

invertptt=0                  ; invert the PTT signal = 1
invertcor=0                  ; invert the COR signal = 1
invertctcss=0                ; invert the CTCSS pin = 1

disablectcss=1               ; only use the cor pin

rxboost=0                    ;


; General config options, with default values shown.
; You should use one section per device, with [general] being used
; for the device.

[general]

piversion2=1                 ; set to 1 for a Raspi B2, 0 for all others
```

```
debug = 0                    ; misc debug flags, default is 0

;rxondelay=0

;------------------------------ JITTER BUFFER CONFIGURATION -------------------------
; jbenable = yes              ; Enables the use of a jitterbuffer on the receiving side of an
                              ; USBRADIO channel. Defaults to "no". An enabled jitterbuffer will
                              ; be used only if the sending side can create and the receiving
                              ; side can not accept jitter. The USBRADIO channel can't accept jitter,
                              ; thus an enabled jitterbuffer on the receive USBRADIO side will always
                              ; be used if the sending side can create jitter.

; jbmaxsize = 200             ; Max length of the jitterbuffer in milliseconds.

; jbresyncthreshold = 1000    ; Jump in the frame timestamps over which the jitterbuffer is
                              ; resynchronized. Useful to improve the quality of the voice, with
                              ; big jumps in/broken timestamps, usualy sent from exotic devices
                              ; and programs. Defaults to 1000.

; jbimpl = fixed              ; Jitterbuffer implementation, used on the receiving side of an USBRADIO
                              ; channel. Two implementations are currenlty available - "fixed"
                              ; (with size always equals to jbmax-size) and "adaptive" (with
                              ; variable size, actually the new jb of IAX2). Defaults to fixed.

; jbtargetextra = 40          ; This option only affects the jb when 'jbimpl = adaptive' is set.
                              ; The option represents the number of milliseconds by which the new
                              ; jitter buffer will pad its size. the default is 40, so without
                              ; modification, the new jitter buffer will set its size to the jitter
                              ; value plus 40 milliseconds. increasing this value may help if your
                              ; network normally has low jitter, but occasionally has spikes.

; jblog = no                  ; Enables jitterbuffer frame logging. Defaults to "no".
;-----------------------------------------------------------------------------
```

Once again, the main concern is the rx pin (corpin) and whether or not it is inverted.

You enable this driver in rpt.conf by setting

```
rxchannel=rpiradio/rpi0
```

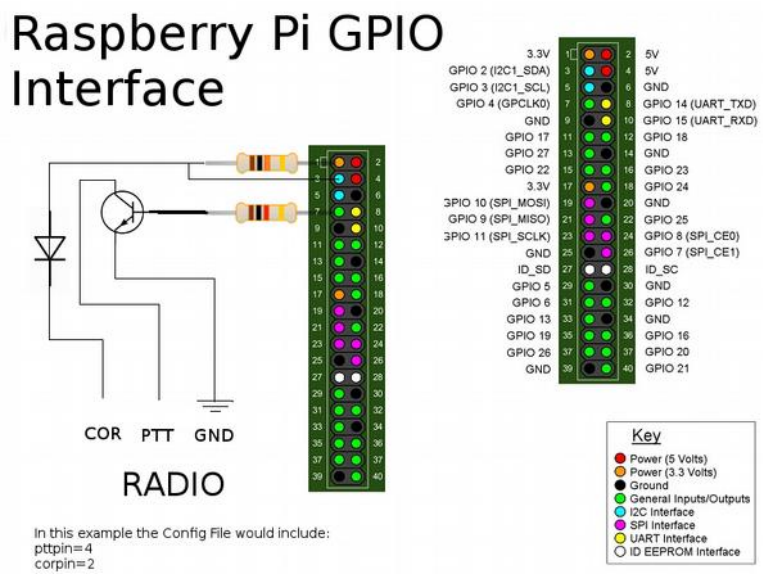
Where rpi0 is the device in the config file.

Example Radio Wiring

This is where we as amateur need to use our noggins a little. Basically, the control pins need to change state, and audio levels need to be correct.

Raspberry Pi GPIO wiring

The description for the RpiRadio driver is exactly what is described here. This applies to ALL of the channel drivers.



The only change that would be required is if your radio pulled high on receive and floated low on standby, you would required another NPN transistor to physically pull the pin to ground.

The fob audio out goes to the audio in in your radio, and the audio out from your radio via the 500k trim pot to the fob mic socket.

Yaesu FT-8800

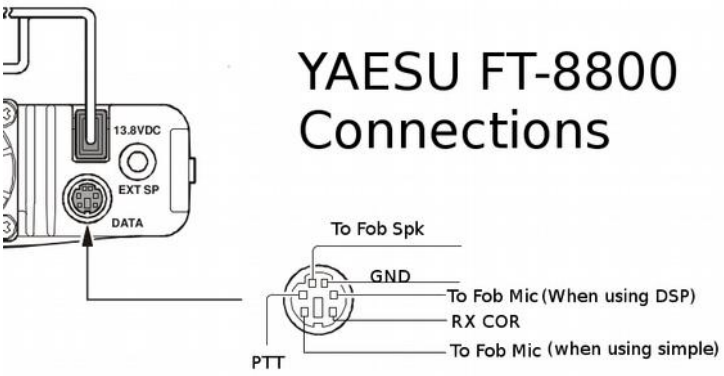
I'll use the Yaesu FT-8800 as an example here, as it has a packet socket which does 9600 and 1200 baud packet.

It has a receiver (COR) pin that goes high on a received signal.

This means that in simpleusb carrierfrom=usb or carrierfrom=rpi. This could also be used in usbradio, except you would use dsp.

In the rpiradio driver, you would leave invertcor=0.

Here is a diagram on how we would hook up the Yaesu FT-8800:



The difference between the Audio out (to fob mic in) in the diagram, is the 1200 baud pin (when using simple) is filtered and limited, and the 9600 baud pin is flat audio for DSP. The filtering and limiting is done in the Pi.

This radio needs *preemphasis=1* set in simpleusb.conf or usbradio.conf.

Phillips/Simoco PRM80

The PRM80 Radio also allows every mode to be used without dismantling the radio.

Pin 3	Ground	
Pin 4	Mic. AF	Input impedance = 470 ohm. sensitivity 40mVrms at 1kHz for 60% deviation. Biassed at +9V. Option for 5mVrms. sensitivity available.
Pin 5	Ground	
Pin 6	Tx audio input	Input impedance = 22k ohm. This input is post clipper, ac-coupled. Sensitivity 200mVrms at 1kHz for 60% deviation.
Pin 7	Mic. mute	Input impedance = 220k ohm. Greater than + 1Vdc will mute Mic. AF. Less than 0.3Vdc to unmute.
Pin 8	PTT	This input is pulled up to + 5.2V via 10k resistor. Grounding this point to less than 1.5Vdc will enable PTT mode. Above 3.5Vdc to disable PTT.
Pin 9	Rx FM out	This output is direct coupled via 10k resistor to unde-emphasised, unmuted audio. 550mVrms for 1kHz/60% Dev. Rx signal. DC output = 4.2V.
Pin 10	Pre vol.audio	This output is post Rx AF processing and is subject to mute. AF level is 550mVrms for 1kHz/60% Dev Rx signal, ac-coupled. This level is not affected by volume control. Source impedance = 470 ohm.
Pin 11	Rx mute out	This output is an open collector pull-down. Active low indicates Rx mute is active. Current sink less than 20mA.

To connect with SimpleUSB the following would be used:

- Pin 3 goes to fob ground
- Pin 4 goes to fob spk out

- Pin 10 goes to fob mic in via a 500k trimpot
- Pin 8 goes to fob ptt
- Pin 11 goes to fob cor input.

To connect with USBRadio:

- Pin 3 goes to gob ground
- Pin 6 goes to fob spk out
- Pin 9 goes to fob mic in via a 500k trimpot
- Pin 8 goes to fob ptt
- Pin 11 goes to fob cor input.

RpiRadio Driver

Depending on which way the RX is pulled on the radio, the input GPIO pins on the Pi need pulling the opposite way via a 100k resistor.

On the Yaesu FT-8800, the RX goes high, so the corpin would need to be pulled to gnd via a 100k resistor.

On the PRM80, it goes low on mute, and open when a signal is present. So, it would need the GPIO pin pulled high to 3.3V via a 100k resistor, so when the mute opens, the gnd is released and it goes high.

Choosing the operating mode

The Operating mode of your node has 4 possible operating modes. This is changed via the rpt.conf file (again in the Main Setup Menu).

The 4 modes are:

- Simplex Mode
- Duplex Mode
- Simplex Link Mode
- Duplex Link Mode

Simplex Mode

This mode is how I envisage most nodes are going to be run. by setting

```
Duplex=1
linktolink=0
```

in the rpt.conf file.

This mode is used on a simplex frequency, and it gives a tail after you let go the button, gives ID's and telemetry beeps to let you know how the connection is going.

Duplex Mode

This mode is how the VKLink repeaters will be running. This is where you have a separate receiver and transmitter and use the node as a repeater controller.

The rpt config file setting:

```
Duplex=2
linktolink=0
```

It behaves as a normal Talk-Through Repeater.

Simplex Link Mode

This mode is used when connecting a gateway to a repeater. Set the config as:

```
Duplex=0
linktolink=0
```

This gives a node that has no tails, no anything, and relies on the repeater to do the Idents etc. These nodes also want to have CTCSS set up on the radio, with CTCSS being generated by the repeater when its receiving a signal, as nobody on the other end of the link wants to listen to repeater tails over the link and then their tails.

Duplex Link Mode

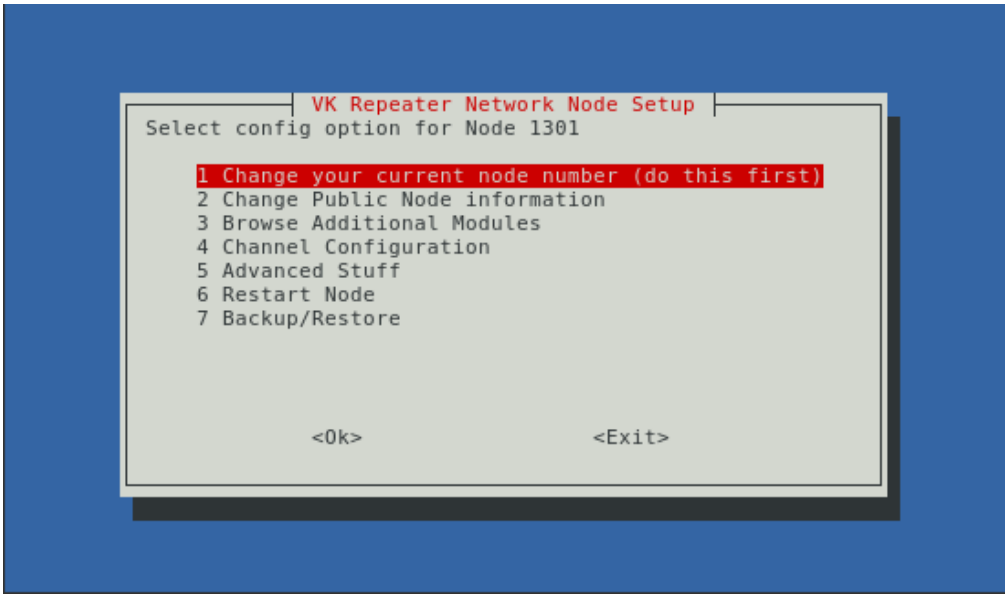
This mode is for linking nodes together over RF. By setting:

```
Duplex=0
linktolink=1
```

Makes the node full duplex without any tones, tails or telemetry. In this mode, it is advisable to change the function tones and set propagate_dtmf=1 so that function commands pass through it to remote nodes.

VKSetup Main Menu

Touching on it briefly before in this manual, the main setup menu allows almost any change to your node. The options will change periodically to keep up with changes to the system.



These are the main options for the node.

If you are not sure what you are doing here, do not do it, as you can wreck things pretty easily.

The first option changes your node number. You should only need to do this once, when you have been given one in the VKLink forum. http://forum.vklink.com.au

Option 2 is all the information that displays on the http://status.vklink.com.au website.

Option 3 allows extra modules to be installed

Option 4 allows the configuration of the channels

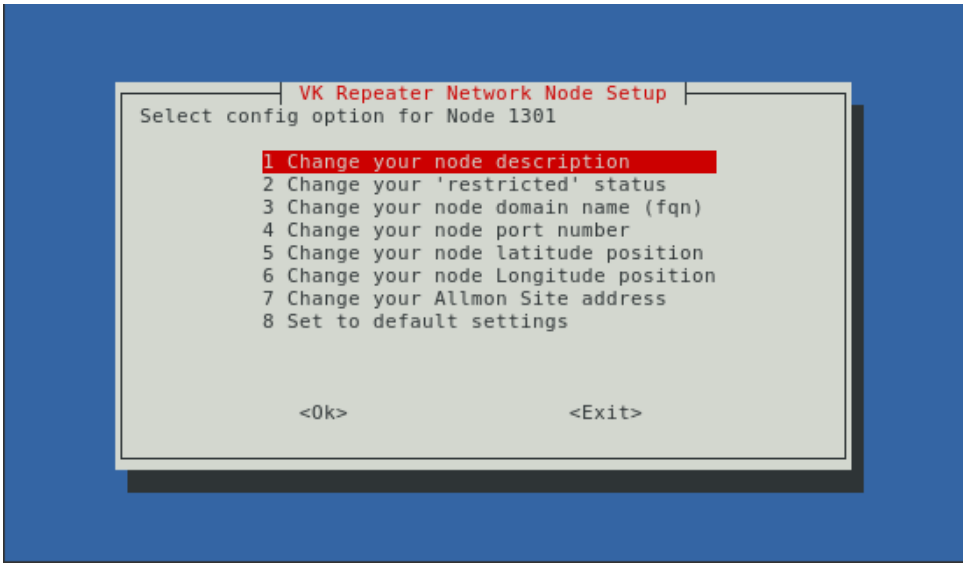
Option 5 is all the Advanced stuff that could get you in trouble!

Option 6 allows you to restart the node (without reboot)

Option 7 is a backup option. This creates a backup on the Vklink server for you to get later on.

Public Information

Here is where you change the information that is seen by the other nodes and http://status.vklink.com.au



1. Change your node description, this would usually be Callsign, Where and Freq. Look at the http://status.vklink.com.au site for examples
2. Restricted Status. If you chose Yes, it goes in the restricted list to show Foundations that it is a 6m or 23cm node. Meaning they should not connect to it.
3. Domain Name. Change this if you have a fqdn, which you will need if you have a dynamic IP address.
4. Port Number, you will not need to change this unless you are running two or more nodes behind a NAT.
5. Latitude position. Enter this in Decimal form (ie -36.323423) for display on the google map.
6. Longitude position. Enter this in Decimal form (ie 146.245345) for display on the google map.
7. Allmon site address. This is the site that loads when somebody clicks you node in the http://status.vklink.com.au webpage
8. Changes your settings back to standard. - Useful when you completely stuff it up.

The Rest of the menus

The rest of the menus will change over time as more drivers get installed, and different directions take shape with the program. Some items do not work. Some are for the old 1.xx image. These will eventually filter out. The menu is dynamic, so any changes are global when they happen.

Changing your callsign in the Nodes On air ID

In the menu system, editing the rpt.conf file allows the changing of your callsign.

Scroll down to *Advanced Stuff* and then scroll down to edit *rpt.conf*

Scroll through that file, and you will find a 2 variables:

```
idrecording = |iid
idtalkover = |iid
```

Change these to reflect your callsign. For example, to set it as VK3SMB, they would appear like:

```
idrecording = |ivk3smb
idtalkover = |ivk3smb
```

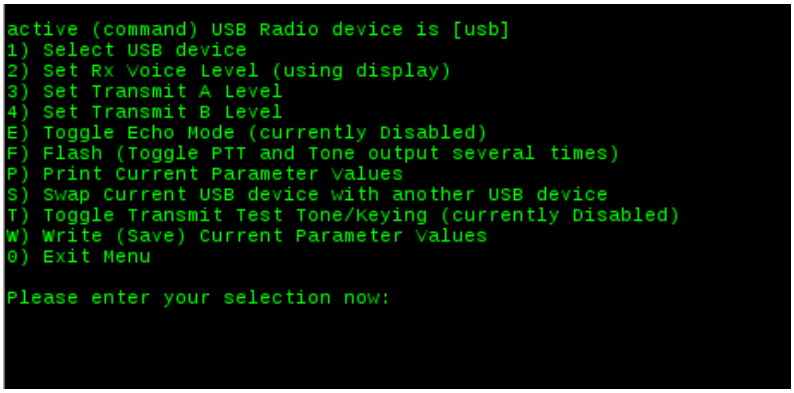
The |*i* needs to stay there as it defines Morse. You can find more information in Making a sound file ID

Setting the Audio Levels

Setting the Audio level for the different channels has a slightly different approach. UsbRadio and SimpleUsb have helper programs to assist with the tuning, and RpiRadio needs you to look at the Asterisk CLI and use some nous.

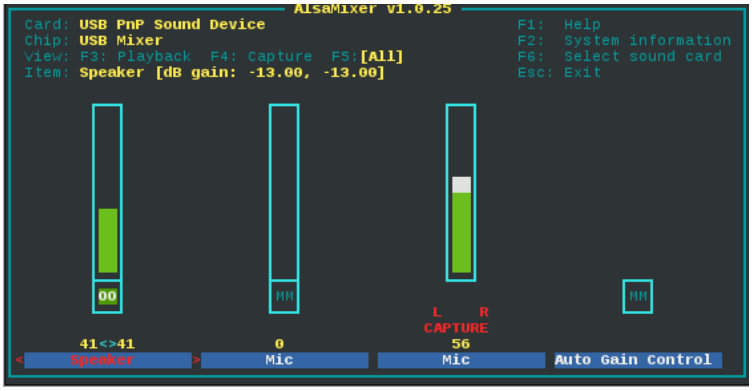
Simple USB

SimpleUSB is just that, very simple. From the Main Setup Menu go into Simple USB Tune. The following will pop up:



1. Generate a 1khz tone at 3khz deviation with a service monitor. Press 2 and follow the instructions. When you have it right, keep pressing enter until you end back at the Menu.
2. With the service monitor in RX, Press T to enable the transmit testing. Press 3 and follow the instructions to set the TX level. If the level is not achievable, Alsamixer needs to be run to set the correct levels. In this case, leave the level at 500.
3. Press W to write the settings and 0 to exit.

If you are unhappy with the levels, from the Main Setup Menu Select Run Alsamixer.



Here you can press F5 to bring up all levels, and use the keyboard to up and down the level. Before you go into the mixer, you can go into the asterisk cli and run the following command:

```
rpt fun {yournodenumber} *989
```

This will put your node into transmit with a 1khz tone at 3khz. You then go back into alsamixer and set the correct level on the service monitor, or what sounds nice from a portable. Press ESC to exit alsamixer and save the current levels

To stop the node from transmitting, go back into Asterisk CLI and run

```
rpt fun {yournodenumber} *989
```

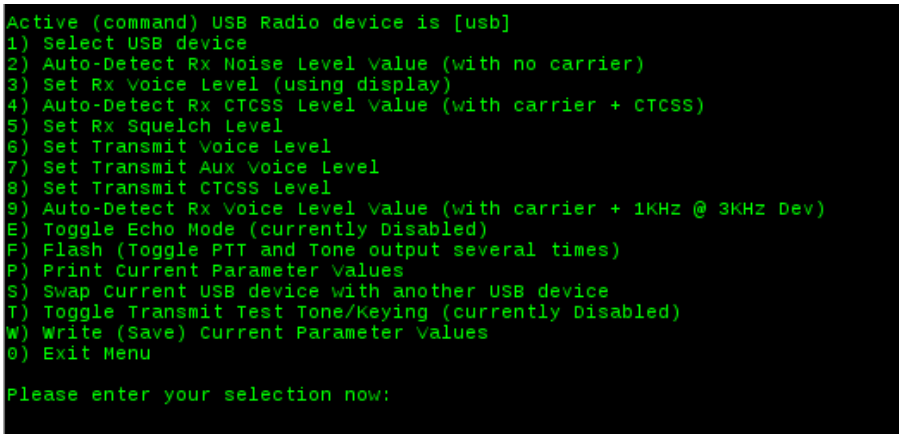
to stop the tone.

Remember it is opposite to what you think. The speaker is the transmit of the radio.

USB Radio

In the Main Setup Menu Select USBRadio tune.

The following will pop up:



Tuning with DSP

Tuning with DSP is easy. The program does most of the work!

1. With the receiver in idle, and no signal, select option 2. If there is not enough noise and the test fails, adjust the 500k pot. When it is successful, go to the next step.
2. Generate a fully quiet signal with a 123hz subtone (or whatever is set in usbradio.conf) at 500-600hz. If a service monitor is not available, use a portable with the deviation set to wideband. Press 4 and let the program adjust itself.
3. Set a 1khz tone at 3khz deviation with the correct subtone fully quiet. If a service monitor is not available, set a portable to narrow deviation and give a whistle. Press 9. keep whistling until it says it has got the correct level (take a deep breath).
4. Put the service monitor on rx. Press T to enable Transmitting. Press 6, follow the instructions to adjust the level to 3khz. If no service monitor is available, listen to a portable until it sounds like a nice signal.
5. Press 8 and set the ctcss tone to 500-600hz on the service monitor. Leave it at a default if you do not have access to a service monitor.
6. To test what it sounds like. Press E to enable Echo Mode. Transmit on a portable with the correct ctcss, and listen to youself and see it if sounds true audio.
7. Press E to disable Echo, Press T to disable test transmit and press W to write the config.
8. Press 0 to exit.

If you cannot whistle long enough for the node to pick up your audio, use option 3 and watch the display while you whistle. you should be able to see where it is going.

Non DSP tuning

Follow the above procedure, however you do not need to set the RX noise level. Unless you are running DSP CTCSS rx is not really possible either.

Rpi Radio

As this is a very basic, very new channel, with bugger all features, tuning it is not as easy as the others. I suggest here that you open a second ssh or putty window to the node to do this procedure. There is no save feature yet, and all the tuning options are run from the Asterisk CLI.

1. In one window, from the Main Setup Menu open the Asterisk CLI. Open AlsaMixer in the other.
2. Set the service monitor to a 1khz tone at 3khz dev or whistle on a narrow dev radio. In the asterisk CLI, type:

```
rpiradio tune rxd
```

3. In the Alsamixer window, Adjust the Capture Mic level to reflect 3khz on the CLI display. When you are happy with this, cut the signal from the service monitor and type

```
exit
```

4. into the CLI window. A little bug I am yet to fix is the letters will repeat themselves in the one spot. This is why we are exiting the CLI.
5. Go back into the CLI
6. Put the Service montor into rx, in the CLI, type:

```
rpt fun {yournodenumber} *989
```

7. Adjust the level of the Speaker in the Alsamixer to reflect 3khz deviation, or what sounds nice on a portable.
8. press {ESC} to exit out of the Alsamixer, type exit to exit out of the CLI.

Dynamic IP addresses

As the security of the system is based predominately on IP addresses for security, Dynamic IP addresses make things a wee bit difficult to handle.

The easiest way to achieve this is to give your node a Domain name with freedns. Now most of Freedns's Domains are registered with spam and all sorts of other rubbish, so they are no good for registering websites etc, but they are fine to use for what we need.

The VK3SMB Demo unit has a Dynamic IP address as it is used on 3G connections for demo purposes.

Through Freedns, I have given it a name of vk3smb.mooo.com

Setting a node name up

1. Go to http://www.freedns.afraid.org
2. Sign up an account
3. Choose a subdomain of one of the free domain names
4. Set an A record to that subdomain
5. Choose Dynamic DNS from the website menu
6. Click Direct URL and it will load up in your browser, most likely with an error.
7. Get the Direct URL setting from the website and copy it from your web browser bar with CTRL-C

Enabling it on the pi

Now that you have a URL for the pi to call at regular intervals, you need to set up a cron-job to call this every 10 minutes and also do it on a reboot.

To set this:

1. Open the Main Setup Menu
2. Go to Advanced stuff
3. choose edit crontab
4. scroll down the bottom of the file and add the following lines:

```
@reboot /usr/bin/wget --delete-after "{yourcopiedurl}" & > /dev/null 2>&1
*/10 * * * * /usr/bin/wget --delete-after "{yourcopiedurl}" & > /dev/null 2>&1
```

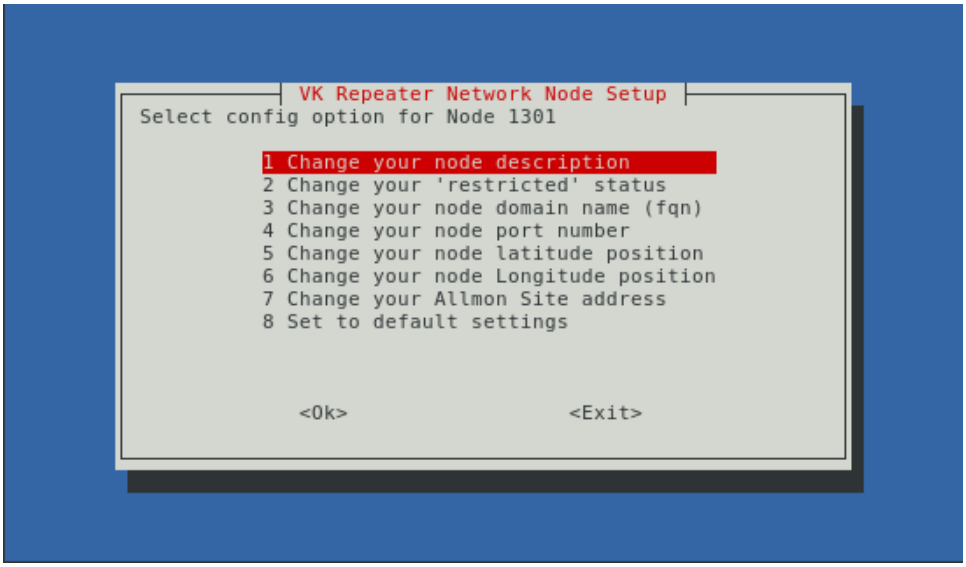
5. Press CTRL X, {enter}, y to save the file

Telling the VKLink server you have a domain name/dynamic IP address

Now the Pi has its own domain, and calls any changes, we have to tell the VKLink server all this, for security reasons, or it will simply boot the pi off the list and nobody will be able to connect to it.

1. Go back into the Main Setup Menu, Select Change Public Node Information.

2. Scroll down to Change your domain name (fqn), press {enter}
3. Enter your domain name, ie vk3smb.mooo.com and press {enter}



The server now knows who your pi is and will keep tabs on it.

Port Forwarding

Given almost every node on the VKlink VK Linking system will be behind a NAT, Port forwarding will need to be set up so connections from the outside can be made.

To set this up correctly, you need to know the operating procedures of your modem, and need to know how to do forwarding. Every modem is different but the basics are:

1. Set a Service. This needs to be port 4569 UDP

   If you are using allmon, you will need to forward port 80 tcp as well

2. Forward incoming connections to the IP address of your node that was set in Initial Setup

3. Most likely, Save your settings and restart the modem.

http://www.portforward.com has instructions to suit most modems.

If you are really stuck the http://forum.vklink.com.au forum is the place to ask questions.

By now You should have received notification in the forum that you have a node number. Time to be Setting up your Node Number.

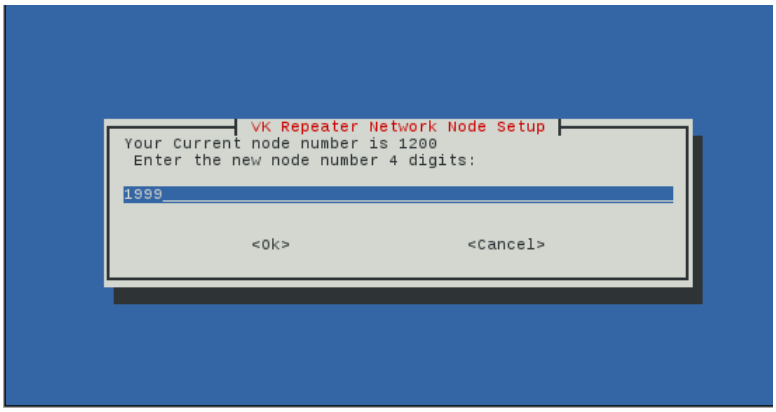
Setting Up your node number

When you have received your node number from the VKlink team, it is a very simple affair to set it:

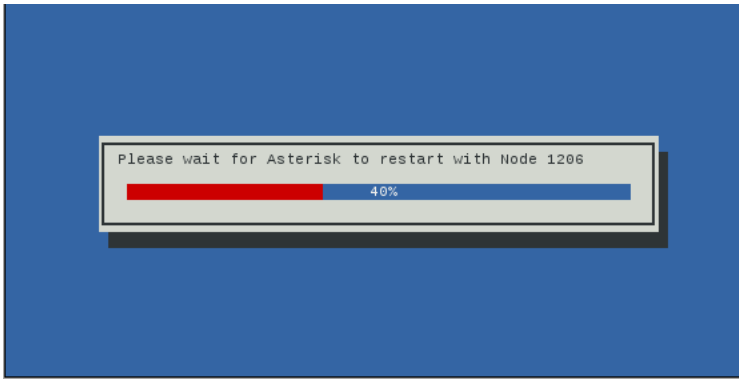
From the Main Setup Menu, select Change your node number, option 1 on the main screen.

Enter the number given to you.

Press {enter}



After a period of time, the node will restart itself,



and if all the other instructions have been followed it will work.

Note

There are three things to note here:

1. The VKlink Team cannot enable your node until it has reported it self for the first time.
2. As the admins for the node system work full time jobs, it may be after 5pm before a node is enabled, and
3. Lastly, you will not be able to connect, or accept connections straight away, as all the other nodes on the system need to be updated to know that your node is on the system. This takes place at 3am every morning. However, if you know the admin of another node, they can run a command on their end to manually force an update, and you will be able to connect to them.

Interfacing to an existing repeater

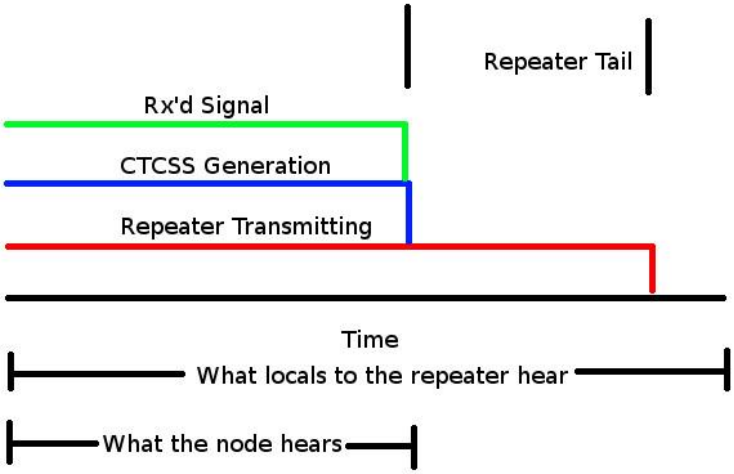
Even though this thing was built to be an actual repeater controller, I realise that most of the nodes out there will be used as a semi-duplex node or repeater gateway type node. Meaning, that it is remote to the repeater.

See Choosing the operating mode to set the node up this way.

To make effective use of this, we need the ability to squash the repeater tails, ID's and courtesy tones on the way through the node.

The simplest way to do this is to get the repeater to generate a CTCSS tone while it is receiving a signal and no other times, and set the remote radio up into tone squelch mode. This means the remote radio will not hear the tones/tail, meaning it will not go across the VK repeater network.

Here is a diagram to try and demonstrate this.



What I'm trying to say here is While there is a received signal on the repeater, the repeater is also generating a CTCSS Tone.

When the received signal stops, so does the CTCSS tone.

What this means is, the other end of the link only hears what the repeater hears, not the tail, ID or courtesy tones.

This makes for much better linking. However this requires co-operation from the repeater custodian.

So by setting the node receiver to Tone Squelch, it only hears what the repeater hears

Using it – DTMF Commands

By Default all commands on the VKLink start with a *. This can be changed by a node owner as described in Advanced Stuff, however, it should be left as default.

There are several commands that is built in as standard to the node system. These include the ability to reset the node, reboot the node and run some test commands.

Default Command List

Every new node image that is downloaded includes the following commands Starting with a * and should end with a #:

1 Disconnect.

2 Monitor.

3 Connect.

4 Run a command.

For example. If I were listening to node 1301 and I wanted to connect to node 1341, I would key *31341. To disconnect, I would dial *11341

Echolink Commands

The echolink driver uses the same first three commands as the VK Link system. The difference, all echolink nodes are padded out to 6 digits with a preceding 3.

For example:

- To connect to the echolink test server (el node 9999), you would dial: *33009999
- To disconnect your would dial *13009999

To connect and disconnect to node 123456, you would dial

- *33123456
- *13123456

IRLP Commands

The IRLP driver uses the same first three commands as the VK Link system. The difference, all IRLP nodes have a preceding 9.

For example:

- To connect to IRLP node number 1234, you would dial *391234
- To disconnect from node number 1234, you would dial *191234

Advanced Stuff

Majority of the Advanced stuff requires you to manually edit files. In the command line, mount the sd card as rw, and then run things as super user:

```
sudo su
mount - / -o remount,rw
```

When you are finished. Reboot the node and it will reset the sd card to read only.

```
sudo reboot
```

Setting up Echolink

Installing and configuring the channel driver

Allowing Echolink on the VKLink node system is as simple as enabling a Channel Driver. You do NOT need to try and install echolink software alongside the VK Link software.

Edit echolink.conf:

```
[el0]
confmode=no                             ; Conference Mode
call=VK3SMB-R                           ; Change this to your
                                        ; echolink callsign including the -r or -l
pwd=xxxxx                               ; Change this to your echolink password!
name=VK Repeater Link Node 1999                 ; Change this to your real name!
qth=Rutherglen Victoria                 ; Change this to your actual QTH!
email=someone@somewhere.net             ; Change this!
maxstns=5                               ; max stations allowed
rtcptimeout=10                          ; rtcp timeout
node=123456                             ; Change this to your echolink node mumber
recfile=/tmp/recorded.gsm
vknode=1999                             ; Change this to your active VK Link node number!
context=radio-secure                    ; extension in extensions.conf
server1=server1.echolink.org
server2=server2.echolink.org            ; echolink servers
server3=server3.echolink.org
freq=438.975                            ; Freq in MHz
tone=123.0                              ; CTCSS Tone (0 for none)
lat=-36.059500                                  ; Latitude in decimal degrees
lon=-146.459375                         ; Longitude in decimal degrees
power=4                                 ; 0=0W, 1=1W, 2=4W,  3=9W, 4=16W, 5=25W, 6=36W,
                                        ; 7=49W,  8=64W,  9=81W  (Power in Watts)
height=1                                ; 0=10' ,1=20',2=40',3=80' ,4=160' ,5=320' ,6=640'
                                        ; ,7=1280',8=2560',9=5120' (AMSL in Feet)
gain=9                                  ; Gain in db (0-9)
dir=0                                   ; 0=omni, 1=45deg, 2=90deg, 3=135deg, 4=180deg,
                                        ; 5=225deg, 6=270deg, 7=315deg, 8=360deg (Direction)
```

You will now need to save and exit the editor (ctrl-x, y, enter).

Now go into Enter Echolink Information in the VKSetup Menu. Answer these questions as they tell the world about your node via the status page.

Check that the module is enabled:

Go into Edit modules.conf and scan through the file for a line that says

```
.…
noload=chan_echolink.so
....
```

And Change it to

```
.…
;noload=chan_echolink.so
....
```

Exit and save the file

Exit the menu system and restart the node via the command:

```
sudo reboot
```

The echolink channel should now be running.

Forwarding ports

As per the documentation on http://www.echolink.org, you need to open two udp ports to your node. Forward port 5198 and 5199 UDP to your vklink node.

Setting Levels

The default levels seem to be pretty good. However to change these you need to edit the rpt.conf file (In advanced options in the Main Setup Menu).

Add these two lines under your VK Link node number square brackets

```
erxgain=-3.0      ; defaults to -3.0 db
etxgain=3.0       ; defaults to 3.0 db.
```

Exit and save the file.

Setting the announce mode

By default the Announcement on the node system is the echolink node number with a preceding 3. This can be changed to do this, Say the Callsign or say both.

Edit the rpt.conf file, and add the following line:

```
eannmode=1
```

Where:

1. is node number
2. is node callsign
3. is both

Make the changes accordingly to what you want and restart the system.

Using the driver

The Commands page has the commands listed to connect to an echolink node from the VK Link node.

For the PC user or true echolink node user, nothing changes connecting to your node.

Setting up IRLP

The BIG BEWARE

While I have thought it to be in the best interest of the Amateur Spirit to include IRLP in this system, IRLP's god and creator, Dave Cameron VE7LTD, has not given an answer. One email has been exchanged between us with his final reply "I'll think about it", and him never answering his "I'll think about it". So your on your own here in that regard. There is information out there on the internet where people have been experimenting, Dave has found out about it and pulled their PGP keys, making their IRLP node innoperable.

The second major issue I see is the security holes that Dave expects his node users to open up into their private networks. I do not own an IRLP node, and with an attitude like that, I never will, but I understand there is something to the tune of 14 ports that need to be opened vs 1 port for us, and 2 for echolink. The main port he expects open is port 22. This is the first place a hacker will try and exploit, as it is the SSH port.

With all that out of the way, I have built and enabled the IRLP channel driver for those who have an existing system with IRLP, so they can merge the systems, cut down on hardware and decrease the power usage. I DO NOT want people who do not have irlp to expect to install the driver and attempt to create a new IRLP node. If you want a new IRLP node, go down the correct channels and order one from Dave, then merge it across. This is not about cutting anyone's throat.

As much as he is pain in the neck, is not very friendly to opposition, and thinks his antiquated hardware can compete with today's systems, it is still his system.

Now thats been explained, onto the good stuff.

Configuring the IRLP channel

Edit irlp.conf:

```
[general]
node=stn1234                ; Change this to your actual IRLP node number!
call=vkxxxx                 ; Change this to your IRLP node callsign
rtcptimeout=10              ; rtcp timeout
localispeakerport=2174      ;
radmode=no                  ;
audioport=2074              ;
context=radio-secure        ; extension in extensions.conf
vknode=1999                 ; Change this to your active VKlink node number!
```

You will now need to save and exit the editor (ctrl-x, y, enter).

Now go into Enter IRLP Information. Answer these questions as they tell the world about your node via the status page.

Check that the module is enabled:

Go into Edit modules.conf and scan through the file for a line that says

```
.…
noload=chan_irlp.so
....
```

And Change it to

```
.…
;noload=chan_irlp.so
....
```

Exit and save the file

Copy all the information (including the hidden files) from the /home/irlp folder on your existing IRLP node to /home/irlp on your VKlink node.

Exit the menu system and restart the node via the command:

```
sudo reboot
```

The IRLP channel should now be running.

Forwarding ports

As per the documentation on the http://www.irlp.net, you need to open ports to your node. Forward the ports to your vklink node.

Setting Levels

The default levels seem to be pretty good. However to change these you need to edit the rpt.conf file (In advanced options in the Main Setup Menu).

Add these two lines under your VK Link node number square brackets

```
irxgain=-10.0      ; defaults to -10.0 db
itxgain=10.0       ; defaults to 10.0 db.
```

Exit and save the file.

Setting the announce mode —Need to test this

By default the Announcement on the node system is the IRLP node number with a preceding 9. This can be changed to do this, Say the Callsign or say both.

Edit the rpt.conf file, and add the following line:

```
irlpann=1
```

Where:

1. is node number
2. is node callsign
3. is both

Make the changes accordingly to what you want and restart the system.

Using the driver

The Commands page has the commands listed to connect to an IRLP node from the VK Link node.

For the IRLP node user, nothing changes connecting to your node.

Iptables firewall

To use IRLP, you will have to open iptables holes to allow traffic. The rules that are needed to be changed are stored in */etc/iptables.rules* and */etc/ip6tables.rules*

Running 2 nodes on 1 Pi

The Pi v2 and up has enough horsepower to run 2 nodes. More testing needs to be done on the drivers to see if the Raspi GPIO's can handle 2 nodes.

In this example, the node number we are using is 1350, and the new Node will be 1351

Edit rpt.conf and duplicate everything between 1350 and functions (abbreviated file):

```
[1350]                                  ; Change this to your assigned node number

rxchannel = Radio/usb                   ; Rx audio/signalling channel
.
.
.
duplex=0                                ; 0 for repeater gateway (use in conjunction with linktolink)
                                        ; 1 for simplex frequency,
                                        ; 2 as a repeater (not connected to one)
;linktolink=1                           ; set to 1 when used as a link full duplex repeater. (no
                                        ; tails,ids etc).

[functions]
.
.
.
```

So that is looks like:

```
[1350]                                  ; Change this to your assigned node number

rxchannel = Radio/usb                   ; Rx audio/signalling channel
.
.
.
duplex=0                                ; 0 for repeater gateway (use in conjunction with linktolink)
                                        ; 1 for simplex frequency,
                                        ; 2 as a repeater (not connected to one)
;linktolink=1                           ; set to 1 when used as a link full duplex repeater. (no
                                        ; tails,ids etc).

[1351]                                  ; Change this to your assigned node number

rxchannel = Radio/usb                   ; Rx audio/signalling channel
.
.
.
duplex=0                                ; 0 for repeater gateway (use in conjunction with linktolink)
                                        ; 1 for simplex frequency,
                                        ; 2 as a repeater (not connected to one)
;linktolink=1                           ; set to 1 when used as a link full duplex repeater. (no
                                        ; tails,ids etc).

[functions]
.
.
.
```

Down the bottom of the file under nodes, you will find a line:

```
1350 = radio@127.0.0.1/1351,NONE
```

Add 1351 to that , keeping the spacing correct so the Main Setup Menu can change it:

```
1350 = radio@127.0.0.1/1350,NONE
1351 = radio@127.0.0.1/1351,NONE
```

Now we need to edit extensions.conf. Find the radio-secure section. This section will have one of two formats depending how long ago vksetup was run:

```
[radio-secure]
exten => 1350,1,rpt,1350
```

or

```
exten => ${NODE},1,rpt,${NODE}
```

Whatever is there needs to be changed to read:

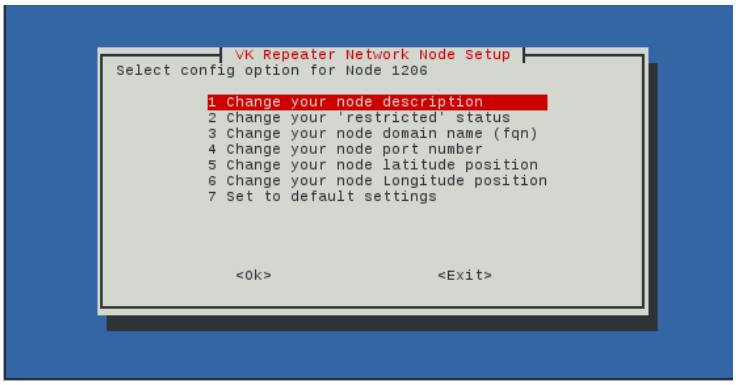
```
exten => 1350,1,rpt,1350
exten => 1351,1,rpt,1351
```

Exit and save and restart the Node with "*sudo reboot*". Any further changes can be made with the Main Setup Menu.

Changing the UDP Port

There is usually no reason to change the standard port on the VK Link. If you run two separate nodes behind a NAT it is mandatory...

The VKSetup Main Menu Has been changed to allow the fully automatic changing of the port.



Simply go into public information and Select option 4.

Setting up a telephone handset

There are times, like with myself, that you are underneath the node, it is connected to a repeater, and you want to use it. Sure you can talk in reverse to the node, except those on the repeater cannot hear you.. Causing an issue that has happened quite often with the IRLP node and Mt Big Ben....

With some software changes, you can connect a SIP phone directly into the your node. This allows you to speak to the node as though you were an external user, without the desense.

Not only that, you can monitor your node at work, or anywhere around the world with the right port forwarding.

Creating a user

A user has to be created to ensure that not just anybody can log into your SIP Phone port.

Create a file called users.conf:

nano /etc/asterisk/users.conf

and insert the following information into that file:

```
[VKNXXX]                                ;Name
registersip = no
host = dynamic
type = peer
username = vknxxx                         ;Username
transfer = no
context = from-handset                    ;Context
cid_number = 0000
hasvoicemail = no
threewaycalling = no
hasdirectory = no
callwaiting = no
hasmanager = no
hasagent = no
hassip = yes                              ;has to be yes
hasiax = no
secret = password                         ;Your Password
nat = yes
canreinvite = no
dtmfmode = rfc2833
insecure = no
pickupgroup = 1
autoprov = yes
linenumber = 1
LINEKEYS = 1
```

SIP setup

Next SIP configurations have to be made. The default port for SIP is 5060. Any port other than that port is recommended so you dont get any chinese hackers trying to make international phone calls through your node. (It happens.....)

nano /etc/asterisk/sip.conf

Add these lines. Noting I have used port 9798 here:

```
[general]
context=default                     ; Default context for incoming calls
allowoverlap=no                     ; Disable overlap dialing support. (Default is yes)
bindport=9798                ; UDP Port to bind to (SIP standard port is 5060)
bindaddr=0.0.0.0             ; IP address to bind to (0.0.0.0 binds to all)
srvlookup=yes                ; Enable DNS SRV lookups on outbound calls
                             ; Note: Asterisk only uses the first host
```

```
                                ; in SRV records
                                ; Disabling DNS SRV lookups disables the
                                ; ability to place SIP calls based on domain
                                ; names to some other SIP users on the Internet
```

The amount of rtp ports need to be limited as well, to save holes in your router.

nano /etc/asterisk/rtp.conf

add or change the file to look like this:

```
[general]
rtpstart=16384
rtpend=16388
```

Creating the route stanza's

As you cannot Dial straight into your node from a SIP phone, a path and conversion needs to be created to allow it to happen. The VKLink channel will not accept SIP, and the SIP phone will not accept IAX.

Edit extensions:

nano /etc/asterisk/extensions.conf

Now noting the context in users.conf add these two sections into extensions.conf:

```
.....…
[dial-in]
exten = s,1,rpt,1999|SX
exten = s,n,Hangup()

[from-handset]                                  ;Context from users.conf
exten = 99,1,Dial(local/s@dial-in,,)
exten = 99,n,hangup()
......
```

So What happens here?

By picking up your SIP phone and dialing 99, Asterisk directs your phone call to the dial-in stanza, and converts the format for us. The dial-in stanza takes your audio and shoves it into the node..

The |SX on the back of the node number (1999), tells VKLink how to use drive the node with the phone. The X means no logging in, and the S means simple control from the phone. The phone cannot connect to any nodes, it listens and transmits to its own node.

Using it

In this example, it works as follows:

- pick up the phone
- dial 99
- to transmit, press *
- to drop ptt, press #
- to disconnect the phone, hang up

Now this works when you are on the same network as the node with your sip phone. What about externally? here's the mods:

Using the SIP phone from an external network

To do this, another few changes need to happen to sip.conf:

nano /etc/asterisk/sip.conf

We need to add our external IP address, the internal network, and a NAT traversal. Add these lines to sip.conf

```
externip=XXX.XXX.XXX.XXX
localnet=192.168.1.0/255.255.255.0         ;your own network address
                                           ;in this case the node address is 192.168.1.161
nat = route
```

We also need to bash open holes in our router to work.....

As a minimum to work, you need, in our example anyway:

port 9798 TCP/UDP to the node internal IP; and

port 16384 UDP to the node internal IP

This varies on all routers, so you are on your own.

What if I dont have a static IP?

Most of us aren't lucky enough to have a static IP address, so we can use the fqdn set up for the node. A simple script has to be written. I have chosen /usr/sipup.sh as the script. so:

nano /usr/sipup.sh

In this file:

```
#!/bin/bash
extip=$(ping $1 -c 1 | grep PING | sed 's/).*//' | sed 's/.*(//')
eval "sed -i 's/externip=.*/externip=$extip/g'" /etc/asterisk/sip.conf
asterisk -rx "module reload"
```

What this does is extract the IP address from the PING command, shove that into the sip.conf file and reload asterisk to accept the change.

Save the file and change it to executable:

chmod +x /usr/sipup.sh

Now we need to insert it into the crontab to run periodically:

crontab -e

Down the bottom insert the following line, inserting your fqdn:

```
.…
*/5 * * * * /usr/sipup.sh [fqdn] & > /dev/null
```

This will mean at most, you wont be able to connect for 5 minutes.

Sip Softphone

Whilst I use a normal SIP house phone to access my node, you can do it with a SIP softphone. In Google Play, there is a nice little piece of open source software called CSip Simple. It is very simple.

1. load the software

2. add your username (this is the one in the square brackets, in our example above it is VKNXXX).
3. enter the IP address of your node, external IP address or fqdn followed by a colon and the sip port. In our example above, it would look like XXX.XXX.XXX.XXX:9798 OR my.domain.name:9798
4. enter the password (secret above); and
5. connect.

Again, use * to PTT and # to drop PTT.

Stand-alone Nodes

This is still very much in development

There are times when a node will need to be set standalone. For example:

- it is a repeater controller without internet
- it is on a private network (defeats the purpose)
- Or on a repeater site where the linking is all in the hut.

There is downsides:

- No updates from the VKLink server
- No node list from the server
- The setup menu will not work
- You need to enter any local nodes manually
- And a bit of ground work is needed

Ground work

This will only work on VKLink version 2 and up.

Before you disconnect your node from the network, you will have to change download the aststart script, and modify the initial startup script:

Download the script:

 # wget http://reporting.vklink.com.au/scripts/aststartpi3.sh -O /usr/aststart.sh

 # chmod +x /usr/aststart.sh

Editing the scripts

Now you have to edit the startup.sh script to invoke this. Here is the original file:

```
#/bin/bash
until /usr/bin/wget --no-check-certificate -q --tries=10 --timeout=20 --spider --delete-after
https://reporting.vklink.com.au; do
    echo "No Internet Connectivity" >&2
    sleep 2
done

/usr/bin/wget --no-check-certificate -N "https://reporting.vklink.com.au/startuppi3.sh" -O
/root/astrun/startup2.sh
chmod +x /root/astrun/startup2.sh
/root/astrun/startup2.sh &
```

Modify it to look like this:

```
#/bin/bash
#until /usr/bin/wget --no-check-certificate -q --tries=10 --timeout=20 --spider --delete-after
https://reporting.vklink.com.au; do
#    echo "No Internet Connectivity" >&2
#    sleep 2
```

```
#done

#/usr/bin/wget --no-check-certificate -N "https://reporting.vklink.com.au/startuppi3.sh" -O /root/astrun/
startup2.sh
#chmod +x /root/astrun/startup2.sh
#/root/astrun/startup2.sh &

sleep 10
#restart asterisk
/usr/sbin/asterisk
sleep 10
killall asterisk
killall asterisk
echo "waiting"
sleep 10
/usr/aststart.sh &
```

Editing rpt.conf

After this is done, edit /etc/asterisk/rpt.conf and add this line to each node:

```
standalone=1
```

Changes in raspi-config

The raspi is set to not start unless a network cable is in. This has to be changed

run:

sudo raspi-config

Got to:

Boot options, press ENTER

Wait for network, press ENTER

and Select NO.

Testing

Without a network cable plugged in, do a reboot and Asterisk should start and stay started after 30 seconds.

Using a proxy (Public NAT workaround)

There has been times where people have wanted to install a node on the blocked side of a public NAT, ie the NAT that sits on the end of your 4g dongle, or at a macca's wifi etc. There is a work around.

What you need is an existing node, ie one that is accessible from the internet, and port TCP port 8654 forwarded from your router to your node. This works the best with a node that has a static IP address.

For this Example node 1998 is the server node, (the one visible from the net), with an IP address of 8.8.8.8, and node 1997 is the remote node (nicknamed remote), the one behind the Public NAT. The password will be 12345678.

VKLink workaround

On the server node, add the following to the bottom of */etc/asterisk/users.conf:*

```
[remote]
type=user
host=dynamic
secret=12345678
context=radio-secure
trunk=yes
callcounter=yes
```

Save that, now add the following to the bottom of */etc/asterisk/iax.conf:*

```
[remote]
type = peer
host = dynamic
context = radio-secure
trunk = no
```

They are the same but different. one is to do with the registering, the other is to do with the connection.

Now, in */etc/asterisk/extensions.conf*, add the following line into the [radio-secure] stanza:

```
exten => 1997,1,Dial(IAX2,remote,1997)
```

Ensure there is no other references to 1997 in the stanza.

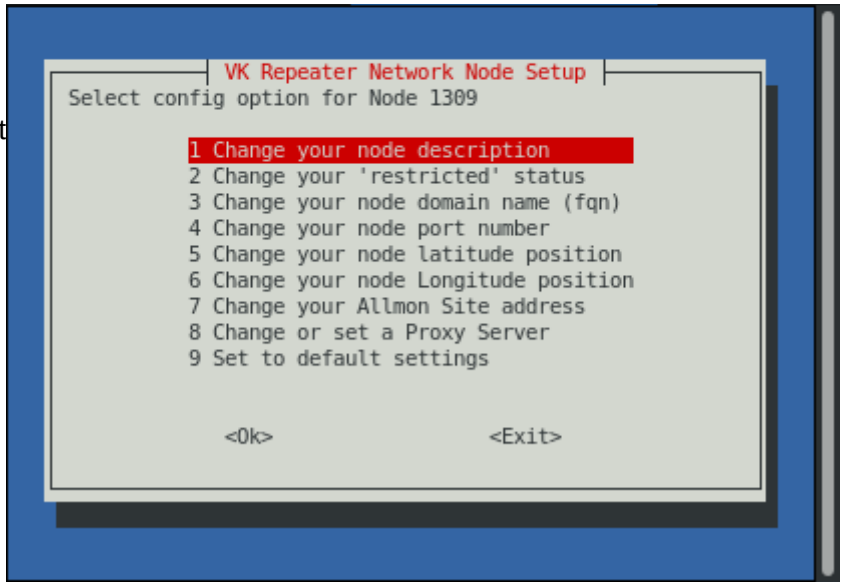
Now. On the remote node, the following needs to be added into the [general] stanza in */etc/asterisk/iax.conf*:

```
register=remote:12345678@8.8.8.8
```

of course, the port can be added after the IP address in the format of 8.8.8.8:port.

Reboot both nodes and one should connect to the other and keep the connection alive, allowing an incoming connection to the remote node via your server node.

You will need to log into VKSetup on 1997, Goto "Public node information" and scroll around until you find "Change or set a Proxy Address", This needs to be changed to the address of your server.



Reversing other Connections

Now that you've got your node going and talking, you may want other services available. For this purpose, We are using ports 1999, 1998 and 1997 on your server. These ports need to be opened on your node if you are using the IPTables firewall (google will help here), and opened to the internet via your router if you want to remotely connect to them. All connections are TCP. We are using SSH to achieve this.

The remote node will still need dynamic DNS sorted for the rest of the VKLink network. Its IP address will still need to match its DNS name.

To allow this to stay up, autossh needs to be installed:

 # sudo apt install autossh

SSH Keys need to be generated and shared between the remote and the server.

For these examples, the Server IP address is 8.8.8.8, and the username pi.

On the remote note. run the following to create the keys and copy them to the server:

```
ssh-keygen
ssh-copy-id -i ~/.ssh/id_rsa.pub pi@8.8.8.8 -p 8654
```

The port 8654 is open on all VKLink nodes for remote support.

SSH Access.

To allow SSH access, the following needs to be entered into */etc/rc.local*, just above *exit 0* (All on one line), then restart rc.local (sudo systemctl restart rc-local.service):

```
sleep 30 && /usr/bin/autossh -N -f -o "ServerAliveInterval=30" -o "PubkeyAuthentication=yes" -o
"PasswordAuthentication=no" -i /root/.ssh/id_rsa -R :1999:localhost:22 pi@8.8.8.8 -p 8654 &
```

What this does is:

- Creates a reverse connection, which links port 1999 on the server to port 22 on the remote node.
- Uses the authorised username on the server, with the open port on the server as the path.
- Pings the connection every 30 seconds to keep it open.

To access ssh on the remote node, you would type the following in a bash shell on the server:

```
ssh remote-node-user@localhost -p 1999
```

Or to access it from a computer that can access your server node:

```
ssh remote-node-user@8.8.8.8 -p 1999
```

Http Access (Using Allmon)

In a very similar fashon to the SSH access, http is done the same way. Add this line just after the SSH line in /etc/rc.local:

```
sleep 30 && /usr/bin/autossh -N -f -o "ServerAliveInterval=30" -o "PubkeyAuthentication=yes" -o
"PasswordAuthentication=no" -i /root/.ssh/id_rsa -R :1998:localhost:80 pi@8.8.8.8 -p 8654 &
```

If your server node had a web browser installed, you would goto *http://localhost:1998*.

But as the nodes do not have a built in GUI, you will have to access it via a different machine. Exactly the same way though: *http://8.8.8.8:1998*

Manager Access (For Allmon on another machine)

You can allow Allmon on another machine access the manager on the remote node in the same way. Add this line to */etc/rc.local*:

```
sleep 30 && /usr/bin/autossh -N -f -o "ServerAliveInterval=30" -o "PubkeyAuthentication=yes" -o
"PasswordAuthentication=no" -i /root/.ssh/id_rsa -R :1997:localhost:5038 pi@8.8.8.8 -p 8654 &
```

This re-directs the manager down the SSH tunnel.

On the server node, you would modify the */var/www/allmon.ini.php* as follows:

```
[1997]
host=localhost:1997
user=admin
passwd=password
menu=yes
hideNodeURL=yes
```

Confusing, yes, the [] brackets is Node 1997, the localhost:1997 is the port.

This will allow Allmon to look at the manager data inside the SSH tunnel.

IP changes on your server

There are times where the server may not be on a static IP address. To solve this problem, a script is made to check the registration status of the node to the server. This script is called from crontab, run say every 10 minutes. As Asterisk does not follow DNS, it uses IP addresses, if a dynamic IP changes itself, Asterisk keeps looking at the resloved IP at the time of the connection.

Create a script:

```
#nano /tmp/ipcorrect.sh
```

in this file put the following:

```
#!/bin/bash

if asterisk -rx "iax2 show registry" | grep -q Unregistered ; then
        asterisk -rx "reload"
fi
```

Save the file. Chmod it to executable and move it to /usr/bin:

```
#chmod +x /tmp/ipcorrect.sh
#mv /tmp/ipcorrect.sh /usr/bin/
```

Now change the crontab to run it every 10 minutes:

```
#crontab -e
```

```
*/10 * * * * /usr/bin/ipcorrect.sh > /dev/null
```

Getting your AllstarLink node number working in VKLink

The first thing you need to to is to go to *http://allstarlink.org* and get a node number.

When you have your node number and password, the following needs to be done:

iax.conf changes

As Allstar runs a different verification process to VKLink, you need to put the register line into your iax.conf. This registers your node with AllstarLink.

Nano */etc/asterisk/iax.conf*

and insert the following line in the [general] stanza:

```
register =>NODENUMBER:PASSWORD@register.allstarlink.org
```

Rpt.conf

you need to duplicate your VKLink node stanza ([1302] for example) in your */etc/asterisk/rpt.conf* configuration file.

Everything from the node heading to the next heading. When you have done this, a couple of changes need to be made. The first is to change the rxchannel to a dummy, and then also tell app_rpt that you want it to report that node to allstarlink.

When you set the rxchannel to dummy, it means any connections by default don't go anywhere. You link your vklink node to your allstar node either with an autoexec script or manually. This means any connections into your allstar node will come out on your VKLink node, and any other VKLink node attached to it.

In you node stanza, change/add the following:

```
rxchannel = dahdi/pseudo
allstar_reporting = 1
```

After these changes, reboot your node. You should see at http://stats.allstarlink.org/nodeinfo.cgi?node=NODENUMBER that your node is now active and reporting. If you check https://allstarlink.org/nodelist.php you should also see your node in green.

If all this is working correctly, you should be able to run *cat /tmp/rpt_extnodes* and see an output that shows all of VKLinks nodes and all of AllstarLinks nodes in one file.

If you have informed the VKLink of your Allstarlink node, it can be also added to the VKLink status page, plus be connected to by other VKLink nodes.

Rpt.conf options

The rpt.conf file has a huge amount of options. Some are more straight forward than others.

Understand Asterisk lingo. A stanza is a section of the config file that starts with square brackets. eg.

[1999]

is your node number stanza

[telemetry]

is the telemetry stanza.

Some items are Boolean.

Boolean answers are either yes or no, 1 or 0, on or off.

Your repeater stanza (node number)

Your repeater stanza [1999] for example is where the options for that repeater go. In here you enter all the parameters needed, and also list other stanza's that it should reference.

These reference stanza's are:

- controlstates stanza
- functions stanza
- link_functions stanza
- macro stanza
- phone_functions stanza
- telemetry stanza
- wait_times stanza
- scheduler stanza, and
- nodes stanza

for example you had two nodes, and you want two sets of functions. A snippet of the rpt.conf file would look similar to this:

```
[1999]
functions=function1
.
.
[1998]
functions=function2
.
.
.



[function1]
dtmf functions go here for 1999
.
.
.

[function2]
dtmf functions go here for 1998
```

This is the same for all the reference stanza's.

Your node stanza options [1999]

althangtime=

This controls the length of the repeater hang time when the alternate hang timer is selected with a control operator function. It is specified in milliseconds.

```
althangtime=4000  ;which equates to 4 seconds.
```

beaconing=

When set to 1 will send the repeater ID at the idtime interval regardless of whether there was repeater activity or not. Boolean

controlstates=

This setting defines the name of the controlstates stanza. Control states are an optional feature which allows groups of control operator commands to be executed all at once.

```
controlstates = your-controlstates-stanza
```

duplex=

The duplex key/value pairs sets the duplex mode for desired radio operation. Duplex mode 2 is the default if none specified.

0. Half duplex with no telemetry tones or hang time. Special Case: Full duplex if linktolink is set to yes. This mode is preferred when interfacing with an external multiport repeater controller. Comment out idrecording and idtalkover to suppress IDs.
1. Half duplex with telemetry tones and hang time. Does not repeat audio. This mode is preferred when interfacing a simplex node.
2. Full Duplex with telemetry tones and hang time. This mode is preferred when interfacing a repeater.
3. Full Duplex with telemetry tones and hang time, but no repeated audio.
4. Full Duplex with telemetry tones and hang time. Repeated audio only when the autopatch is down.

endchar=

This setting allows the end character used by some control functions to be changed. By default this is a #. The endchar value must not be the same as the funcchar default or overridden value.

erxgain=

Echolink receive gain adjustment in +/- db-volts. Used to balance Echolink recieve audio on an app_rpt node.

etxgain=

Echolink transmit gain adjustment in +/- db-volts. Used to balance Echolink transmit audio on an app_rpt node.

funchar=

This setting can be used to change the default function starting character of * to something else. Please note that the new value chosen must not be the same as the default or overridden value for endchar=.

functions=

This points to the the functions stanza

```
functions = your-functions-stanza
```

hangtime=

This controls the length of the repeater hang time. It is specified in milliseconds.

```
hangtime=4000  ;which equates to 4 seconds.
```

holdofftelem=

This node stanza configuration key forces all telemetry to be held off until a local user on the receiver or a remote user over a link unkeys. There is one exception to this behavior: When an ID needs to be sent and there is activity coming from a linked node. Boolean.

idrecording=

The identifier message is stored in the node stanza using the idrecording key. It can be changed to a different call sign by changing the value to something different. The value can be either a morse code identification string prefixed with |i, or the name of a sound file containing a voice identification message. When using a sound file, the default path for the sound file is /var/lib/asterisk/sounds.

```
idrecording = |iwa6zft/r   ; Morse Code ID
```

or

```
idrecording = |myid   ; Voice ID
```

Note: ID recording files must have extension gsm,ulaw,pcm, or wav. The extension is left off when it is defined as the example shows above. File extensions are used by Asterisk to determine how to decode the file. All ID recording files should be sampled at 8KHz. See Making a sound file ID

idtalkover=

The ID talkover message is stored in the node stanza using the idtalkover setting. The purpose of idtalkover is to specify an alternate ID to use when the ID must be sent over the top of a user transmission, This can be a shorter voice ID or an ID in morse code. The value can be either a morse code identification string prefixed with |i, or the name of a sound file containing a voice identification message. When using a sound file, the default path for the sound file is /var/lib/asterisk/sounds. Same usage as idrecording. Also see Making a sound file ID

inxlat=

The inxlat setting allows complete remapping of the funcchar and endchar digits to different digits or digit sequences. inxlat acts on the digits received by the radio receiver on the node.

Format: inxlat = funchars,endchars,passchars,dialtone

where:

- funcchars is the digit or digit sequence to replace funcchar
- endchars is the digit or digit sequence to replace endchar
- passchars are the digits to pass through (can be used to block certain digits)
- dialtone is optional. Set to Y for dial tone on successful funcchars.

link_functions=

This points to the link_functions stanza.

The link_functions setting directs the node to use a particular function stanza for functions dialed by users accessing the node via a link from another node. The traditional default is to point it to a function stanza named "functions".

linkmongain=

Link Monitor Gain adjusts the audio level of monitored nodes when a signal from another node or the local receiver is received. If linkmongain is set to a negative number the monitored audio will decrease by the set amount in db. If linkmongain set to a positive number monitored audio will increase by the set amount in db. The value of linkmongain is in db. The default value is 0 db.

linktolink=

When operating in duplex mode 0, this forces the radio interface to operate in full duplex mode, but keeps all the other duplex mode 0 semantics. This is used when a radio interface is connected to a multiport analog repeater controller. The linktolink= key can take two values: yes or no.

linkunkeyct=

The linkunkeyct setting selects the courtesy tone to send when a connected remote node unkeys. The default is no courtesy tone when a remote node unkeys. the ct tones are defined in the telemetry stanza.

```
linkunkeyct = ct8
```

macro=

The marco key/value points to by the macro stanza.

```
macro = your-macro-stanza
```

nounkeyct=

The nounkeyct node stanza key completely disables the courtesy tone. This is useful for eliminating TX time in applications using simplex uplinks to repeaters on the repeater pair itself. This practice is **strongly** discouraged. Boolean.

politeid=

The politeid setting specified the number of milliseconds prior to the end of the id cycle where the controller will attempt to play the ID in the tail when a user unkeys. If the controller does not get a chance to send the ID in the tail, the ID will be played over the top of the user transmission. Optional, default 30000.

propagate_dtmf=

Takes either yes or no. When set to yes, DTMF will be regenerated from out-of-band signalling or from from the receiver dtmf decoder whenever a function start character is NOT detected and command decoding has not begun. When set to no, no tones will be regenerated. The default for this setting is no. This setting is meant to be used in conjunction with linktolink, inxlat, and outxlat when interfacing a radio port to a multiport analog repeater controller on an RF-linked system. Boolean.

remotect=

The remotect setting allows the selection of the remote linked courtesy tone so that the users can tell there is a remote base connected locally.

```
remotect = ct3
```

rxchannel=

The rxchannel key/value pair selects the radio interface. There must be one (and only one) rxchannel per node. The selections are:

Value	**Description**
dahdi/pseudo	No Radio, used for hubs
simpleusb/usb0	SimpleUsb - main driver we use
radio/usb0	USBRadio - full DSP etc
rpiradio/rpi0	rpiradio - experimental alsa driver
USRP/127.0.0.1:34001:32001	GNU Radio interface USRP - could be made into DMR interfacing!!!
voter/vote0	Voter usage - contact VK7HH

The format is channeldriver/section-in-channel-driver

schedule=

The schedule key/value points to by the scheduler stanza.

```
schedule = your-schedule-stanza
```

sleeptime=

This sets the inactivity period in seconds for no signal on the repeater receiver before the system goes to sleep. Time in seconds.

startup_macro=

The startup_macro is executed once on system startup. Each node can have one startup macro defined in its node stanza.

```
startup_macro = *31000*31001*31002   ; Connect to 1000, 1001 and 1002 at startup
```

tailmessagelist=

The tailmessagelist setting allows a comma separated list of audio files to be specified for the tail message function. The tail messages will rotate from one to the next until the end of the list is reached at which point the first message in the list will be selected. If no absolute path name is specified, the directory var/lib/asterisk/sounds will be searched for the sound file. The file extension should be omitted.

```
tailmessagelist = welcome,clubmeeting,wx   ; rotate 3 tail messages
```

Note: ID recording files must have extension gsm,ulaw,pcm, or wav. The extension is left off when it is defined as the example shows above. File extensions are used by Asterisk to determine how to decode the file. All ID recording files should be sampled at 8KHz.

tailmessagetime=

This sets the amount of time in milliseconds between tail messages. Tail Messages are played when a user unkeys on the repeater input at the point where the hang timer expires after the courtesy tone is sent.

tailsquashedtime=

If a tail message is "squashed" by a user keying up over the top of it, a separate time value can be loaded to make the tail message be retried at a shorter time interval than the standard tailmessagetime= setting. The tailsquashedtime= setting takes a value in milliseconds.

telemdefault=

Controls the voice telemetry.

0. off
1. on
2. timed on status before off

0 is useful for when there is a node on a repeater that people don't need to know its there. Default is 2.

totime=

This defines the time out timer interval. The value is in milliseconds.

unlinkedct=

The unlinkedct setting selects the courtesy tone to be used when the system has no remote nodes connected and is operating as a standalone repeater.

```
unlinkedct = ct2
```

wait-times=

Wait-times points to the wait_times stanza.

controlstates stanza

There are several predefined nmemonics (keywords) used in the control state stanza to enable and disable the various features of the controller. These nmemonics correspond to the control operator command to be executed and most of these are the same groups of letters sent back when a single control operator command is executed on the controller.

Nmemonic	Description	COP Method
rptena	Repeater Enable	2
rptdis	Repeater Disable	3
totena	Timeout Timer Enable	7
totdis	Timeout Timer Disable	8
apena	Autopatch Enable	9
apdis	Autopatch Disable	10
lnkena	Link Enable	11
lnkdis	Link Disable	12
skena	Scheduler Enable	15
skdis	Scheduler Disable	16
ufena	User Functions Enable	17
ufdis	User Functions Disable	18
atena	Alternate Hangtime Enable	19
atdis	Alternate Hangtime Disable	20
noice	No Incoming Connections Enable	49
noicd	No Incoming Connections Disable	50
slpen	Sleep Mode Enable	51
slpds	Sleep Mode Disable	52

functions stanza

This stanza is where the power of control comes in. all DTMF functions are set here. The format for the functions are:

```
;dtmfcode=control-function,control-function-id

;example dtmf *91 would make permalink connection

91=ilink,13
```

This is also where you would tell app_rpt a dmtf command to engage a macro:

example:

```
5=macro,1
```

This would run macro 1 in the macro stanza.

List of control-functions

ilink (control-function) commands listed by control-function-id

These are added {dmtf}=ilink,{number}

1 - Disconnect specified link

2 - Connect specified link — monitor only

3 - Connect specified link — tranceive

4 - Enter command mode on specified link

5 - System status

6 - Disconnect all links

7 - Last Node to Key Up

8 - Connect specified link — local monitor only

9 - Send Text Message (9,<destnodeno or 0 (for all)>,Message Text, etc.

10 - Disconnect all RANGER links (except permalinks)

11 - Disconnect a previously permanently connected link

12 - Permanently connect specified link — monitor only

13 - Permanently connect specified link — tranceive

15 - Full system status (all nodes)

16 - Reconnect links disconnected with "disconnect all links"

17 - MDC test (for diag purposes)

18 - Permanently Connect specified link — local monitor only

200 thru 215 - (Send DTMF 0-9,*,#,A-D) (200=0, 201=1, 210=*, etc)

Status commands

These are added {dtmf}=status,{number}

1 - Force ID (global)

2 - Give Time of Day (global)

3 - Give software Version (global)

4 - Give GPS location info

5 - Last (dtmf) user

11 - Force ID (local only)

12 - Give Time of Day (local only)

cop commands

These are added {dtmf}=cop,{number}(and extra values if needed separated via comma's)

1 - System warm boot

2 - System enable

3 - System disable

4 - Test Tone On/Off

5 - Dump System Variables on Console (debug)

6 - PTT (phone mode only)

7 - Time out timer enable

8 - Time out timer disable

9 - Autopatch enable

10 - Autopatch disable

11 - Link enable

12 - Link disable

13 - Query System State

14 - Change System State

15 - Scheduler Enable

16 - Scheduler Disable

17 - User functions (time, id, etc) enable

18 - User functions (time, id, etc) disable

19 - Select alternate hang timer

20 - Select standard hang timer

21 - Enable Parrot Mode

22 - Disable Parrot Mode

23 - Birdbath (Current Parrot Cleanup/Flush)

24 - Flush all telemetry

25 - Query last node un-keyed

26 - Query all nodes keyed/unkeyed

32 - Touchtone pad test: command + Digit string + # to playback all digits pressed

33 - Local Telemetry Output Enable

34 - Local Telemetry Output Disable

35 - Local Telemetry Output on Demand

36 - Foreign Link Local Output Path Enable

37 - Foreign Link Local Output Path Disable

38 - Foreign Link Local Output Path Follows Local Telemetry

39 - Foreign Link Local Output Path on Demand

40 - IRLP announce Enable

41 - IRLP announce Disable

42 - Echolink announce node # only

43 - Echolink announce node Callsign only

44 - Echolink announce node # & Callsign

45 - Link Activity timer enable

46 - Link Activity timer disable

47 - Reset "Link Config Changed" Flag

48 - Send Page Tone (Tone specs separated by parenthesis)

49 - Disable incoming connections (control state noice)

50 - Enable incoming connections (control state noicd)

51 - Enable sleep mode

52 - Disable sleep mode

53 - Wake up from sleep

54 - Go to sleep

55 - Parrot Once if parrot mode is disabled

56 - Rx CTCSS Enable

57 - Rx CTCSS Disable

58 - Tx CTCSS On Input only Enable

59 - Tx CTCSS On Input only Disable

61 - Send Message to USB to control GPIO pins (cop,61,GPIO1=0,GPIO4=1.....)

62 - Send Message to USB to control GPIO pins, quietly (cop,62,GPIO1=0,GPIO4=1.....)

63 - Send pre-configred APRSTT notification (cop,63,CALL,OVERLAYCHR)

64 - Send pre-configred APRSTT notification, quietly (cop,64,CALL,OVERLAYCHR)

65 - Send POCSAG page (equipped channel types only)

Shell Commands

These are added {dtmf}=cmd,{command including path}

For example to run a script called fred.sh, with dtmf *68, it would look like:

68=cmd,/bin/sh {pathtoscript}/fred.sh

link_functions stanza

see functions stanza

macro stanza

macro's are listed the same as functions, with the format:

dtmf = macro-command

example:

```
1=*31998*31997#
```

using the macro example in the functions stanza, you would DTMF *51 to run this macro.

phone_functions stanza

see functions stanza

telemetry stanza

This stanza is named by the telemetry= key/value pair. Telemetry entries can be shared across all nodes on the Asterisk/app_rpt server, or defined for each node. Can be a tone sequence, morse string, or a file as follows:

- |t - Tone escape sequence:
  - Tone sequences consist of 1 or more 4-tuple entries (freq1, freq2, duration, amplitude). Single frequencies are created by setting freq1 or freq2 to zero.
- |m - Morse escape sequence:
  - Sends Morse code at the **telemetry amplitude and telemetry frequency** as defined in the [morse] section. Follow with an alphanumeric string.
- |i - Morse ID escape sequence:
  - Sends Morse code at the **ID amplitude and ID frequency** as defined in the [morse] section. Follow with an alphanumeric string.

- Path to sound file:
    - Specify the path to a sound file on the server. Do not include file extension.

The following example is in the rpt.conf file by default:

```
[telemetry]
ct1=|t(350,0,100,2048)(500,0,100,2048)(660,0,100,2048)
ct2=|t(660,880,150,2048)
ct3=|t(440,0,150,4096)
ct4=|t(550,0,150,2048)
ct4=|t(2475,0,250,768)
ct5=|t(660,0,150,2048)
ct6=|t(880,0,150,2048)
ct7=|t(660,440,150,2048)
ct8=|t(700,1100,150,2048)
ct9=|t(1633,0,50,1000)(0,0,30,0)(1209,0,50,1000);
;remotetx=|t(1633,0,50,3000)(0,0,80,0)(1209,0,50,3000);
remotetx=|t(880,0,150,2048)
remotemon=|t(1209,0,50,2048)
cmdmode=|t(900,903,200,2048)
functcomplete=|t(1000,0,100,2048)(0,0,100,0)(1000,0,100,2048)
patchup=rpt/callproceeding
patchdown=rpt/callterminated

What the numbers mean,
 (000,000,010,000)
   |   |   |   |-------amplitude
   |   |   |-------------duration
   |   |-------------------Tone 2
   |-------------------------Tone 1
```

wait_times stanza

The wait time stanza is used to set delay time between various node actions and their response. Values are in milliseconds.

This is the code from rpt.conf

```
[wait-times]
telemwait=2000                          ; Time to wait before sending most telemetry
idwait=500                              ; Time to wait before starting ID
unkeywait=2000                          ; Time to wait after unkey before sending CT's and link telemetry
calltermwait=2000                       ; Time to wait before announcing "call terminated"
```

scheduler stanza

The scheduler is used to execute commands at a particular time. the format is in crontab format.

```
[your-schedule-stanza]
;dtmf_function =  m h dom mon dow  ; ala cron, star is implied
2 = 00 00 * * *   ; at midnight every day, execute DTMF code 2.
```

morse stanza

Morse code parameters, these are common to all nodes

example in the rpt.conf file:

```
[morse]
speed = 20            ; Approximate speed in WPM
frequency = 900       ; Morse Telemetry Frequency
amplitude = 4096      ; Morse Telemetry Amplitude
idfrequency = 746     ; Morse ID Frequency
idamplitude = 768     ; Morse ID Amplitude
```

nodes stanza

The [nodes] stanza is a list of nodes, their IP addresses, port and "NONE" . The nodes stanza is used to identify which node is mapped to which Internet call and to determine the destination to send the call to. Only place a definition for your local nodes, and private (off VKLink) nodes or nodes behind the same NAT router here.

Example:

```
[nodes]
1999 = radio@127.0.0.1/1999,NONE              ; This node
1998 = radio@host.domain.com/1998,NONE        ; Private node on another server
```

Miscellanous

Making a sound file ID

Instead of using morse to identify your node, you can combine several sound files to make a voice Ident. To make the callsign VK3VS for example your would type the following in a bash shell:

```
Cat /var/lib/asterisk/sounds/letters/v.gsm /var/lib/asterisk/sounds/letters/k.gsm
/var/lib/asterisk/sounds/digits/3.gsm /var/lib/asterisk/sounds/letters/v.gsm
/var/lib/asterisk/sounds/letters/s.gsm > /var/lib/asterisk/sounds/rpt/vk3vs.gsm
```

In Your repeater stanza (node number), idrecording= and idtalkover= would now be

```
idrecording=rpt/vk3vs
```

Allmon Password

By default, the node image does not include a password for Allmon. This password is used in the web interface to connect and control your node. To add a password, open a bash shell and run the following commands:

```
cd /var/www
sudo rm .htaccess
sudo htpasswd -c .htpasswd admin
```

Being "admin" is the username you want.

If you cannot log into allmon, retype the commands and add *-d* after the *-c*.

Useful Scripts

As usual with scripts you want to use. it is a good idea to store them in /usr/bin and to change the permissions to executable (chmod +x).

Permalink fix

make a config file in /etc/asterisk called perms.conf

```
nano /etc/asterisk/perms.conf
```

add one node number you wish to stay connected to per line:

```
1998
1997
1996
```

now make the program script:

```
nano /usr/bin/perms.sh
```

and enter the following:

```
#!/bin/bash
NODE=1999                               # your node number
permini=/etc/asterisk/perms.conf        # your perm list file

#see which nodes are connected to us
nodes_conn=$(/usr/sbin/asterisk -rx "rpt nodes $NODE" | sed 's/[^0-9,]*//g' | sed '/^$/d')

#pull the nodes needed to be connected to out of an ini script
nodes_need=$(cat $permini)


#check each node is connected individually
set $nodes_need

for word in $nodes_need

do

connected=$(echo $nodes_conn | grep -ic "$word")
        if [ $connected  -eq 1 ]
        then
        echo $word is connected
        else
        /usr/sbin/asterisk -rx "rpt cmd  $NODE cop 34 0"                #turn off telem
        /usr/sbin/asterisk -rx "rpt fun $NODE *3$word"                  #reconnect node
        /usr/sbin/asterisk -rx "rpt cmd  $NODE cop 33 0"                #turn telem back on
        fi

done

exit 0
```

After that, change the permissions:

```
chmod +x /usr/bin/perms.sh
```

Then put it into the crontab:

```
crontab -e
```

down the bottom enter:

```
00 * * * * /usr/bin/perms.sh > /dev/null 2>&1       #runs on the hour every hour
```

Checking COR is working as it should

By VK7DB

Run this code and it should show you on screen if there is a signal on the COR pin or not.

*Only works with one node in the system.

```
#/bin/bash

clear
node=$(asterisk -rx "rpt localnodes" | sed 's/[^0-9]*//g' | sed 's/ //g' | sed '/^$/d')
    if [ $node = 0 ]
        then
        whiptail --title "VK Repeater Network" --msgbox "Asterisk is not running, you need to wait until
Asterisk has started to run this program" 9 43
        exit 0
    fi

while :
do
asterisk -rx "rpt stats $node" | grep "Signal on input"
sleep 0.5
clear
done
```

put this into a script and change the permissions. press *ctrl-c* to exit

Foundation Licence Holders

Foundation licencees

As the ACMA does not allow foundation guys and girls to make and use an RF-Node system, I thought I'd share an idea I saw on http://crompton.com/hamradio/USBFOB_without_radio/.

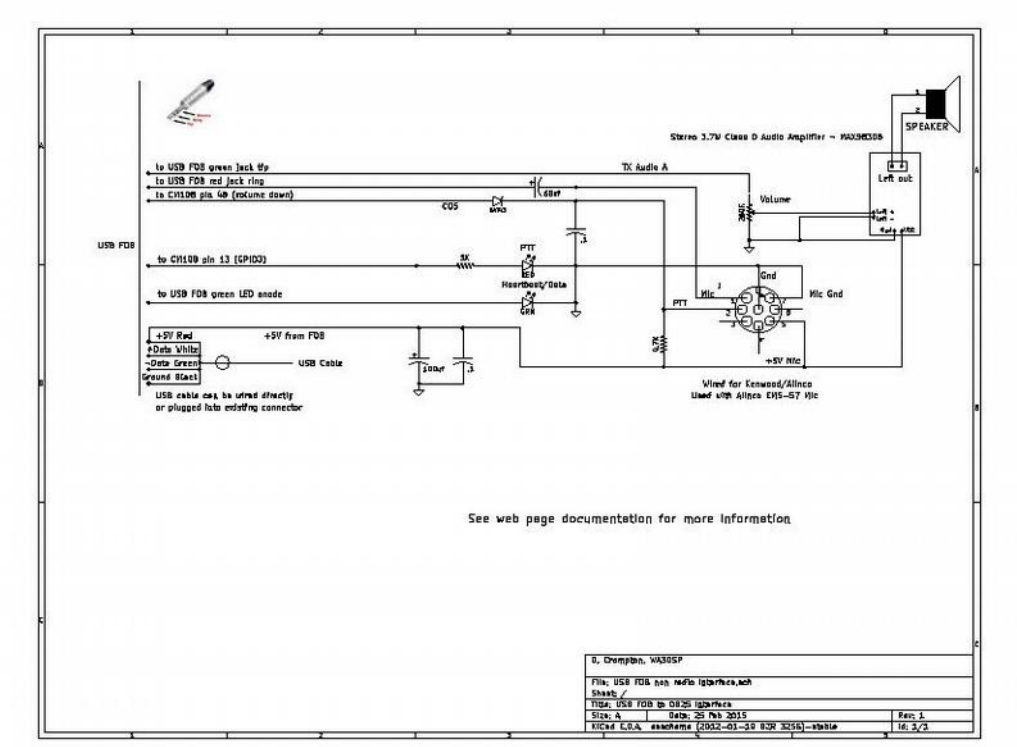
It involves:

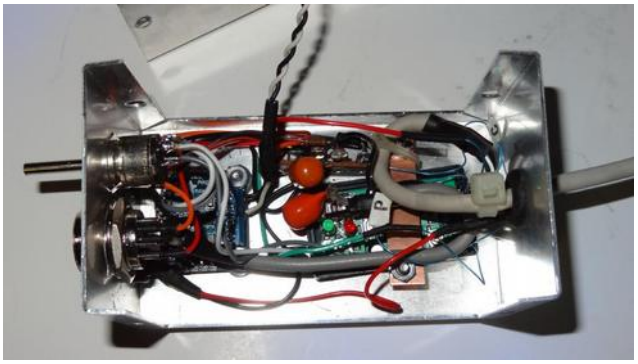
- Setting up a node as per this Manual
- Installing an amplifier for audio out of the node
- Installing a DTMF Microphone as PTT (RX cor)and audio into the node.

The pictures and diagrams here are courtesy of the hamvoip page, and as I get the chance to make one from Australian Bits, I will change the images and schematic.

The Alinco Mic runs from 5V so it can be powered from the USB socket of the Pi. You only need to run simpleusb for this project.

As our system can utilise the GPIO's of the Raspi, a lot of the hardware in these photos is not needed.





Comparison Chart of Linking systems.

	VK Link	Echolink	IRLP
Hardware	Raspi + Dongle	PC Computer	PC or Raspi + Proprietary Board
Cost DIY new	< 100	500ish (new PC)	✗
Cost buy new	220	500ish (new PC)	250 + new PC
Network Ports required	1	2	23 or 24
Internet Security	IAX2 Authentication, Iptables with fail2ban	Relies on Windows Firewall	No firewall on nodes, no DoS protection, Default SSH port, PGP Authentication on Network
SIP/IAX Backbone	✓	✗	✗
<5W power consumption	✓	✗	✗ PC, ✓ Rpi
Full Duplex Audio	✓	✗	✓
Audio Quality	Excellent	Poor	Average
Full DSP	✓	✗	✗
Can be used as a repeater controller	✓	✗	✓
Inter-system connectivity	✓	✗	✗
Multiple Connections	✓	✓	✗
Permanent Connections	✓	✗	✓
Allows Off-Air Linking	✓	✗	✗
Random Connections	✓	✓	✗
Web Control Interface	✓	✗	✓
PC Control Application	✓	✓	✗
Open Source Codecs	✓	✓	✓
Open Source Software	✓	✗	✗
Allows Software Experimentation	✓✓✓	✗	✗
Allows Script Experimentation	✓	✗	✓
Allows Network Experimentation	✓	✗	✗
Australian Designed	✓	✗	✗
Australian Only Network	✓	✗	✗
Allows remote control	✓	✓	✓
Allows remote shutdown on fault	✓	✗	✓
Documentation	In progress	Excellent	OK

Known Bugs

1. permalink connections do not work as they should.
    1. It appears that the IAX channel is cancelling the permalink. this either needs to be looked into or, fixed with a script
2. dtmf propagate does not work as it should either..

Dont's

There has been some users who thought I would be a good idea to use Raspi-update or similar. This is a bad Idea. This script updates the kernel on the raspberry Pi.

The Kernel has been streamlined to suit VKLink. Also, the Dahdi Timing modules are build by dahdi on installation to suit the kernel.

If you update the kernel, you lose the streamlining of the kernel, and you WILL break the dahdi modules, rendering Asterisk, and hence VKLink unusable.

Be Warned. Dont do it.