



TNC-Pi Assembly Instructions & Operating Tips

By John Hansen, W2FS

Portions by

John Wiseman, G8BPQ – Linux configuration

Jim Whiteside, M0HPJ – Xastir

Paul Fischer, KC9RGZ – Headless iGate

Document editing support by

Ed Slingland, N2WD

Website

<http://tnc-x.com/TNCPi.htm>

Contents

Introduction	3
Assembly Instructions	4
Parts List	4
Assembly	6
Parts Layout.....	10
Connecting the Radio	11
Adjust transmit audio output.....	11
Set the transmit delay	11
Configuring the Raspberry Pi	12
Keyboard to Keyboard Connections	12
Support	14
Application Notes	14
Using Xastir	14
Update the package list	14
Then install Xastir	14
Kill <code>kissattach</code>	15
Start Xastir	15
Running Xastir.....	15
More information	16
Configuring TNC-Pi for Use with the I2C Protocol	16
Setup	16
Get and set parameters	16
Running Applications other than LinBPQ with TNC-Pi in I2C mode.....	17
Steps to create an APRS receive-only igate using the Raspberry Pi and TNC-Pi	18
Equipment needed	18
Instructions.....	19

Introduction

Thank you for purchasing a TNC-Pi: TNC-X for Raspberry Pi. This assembly/operating manual is necessarily very preliminary as the physical layout of the TNC-Pi may change slightly in the coming weeks. Also, since TNC-Pi is a brand new product it will take some time to accumulate more operating experience with it.

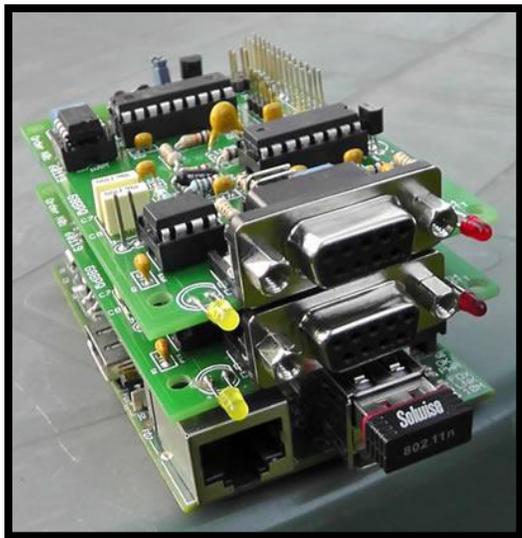


Figure 1: Two TNC-Pi's stacked on one Raspberry Pi

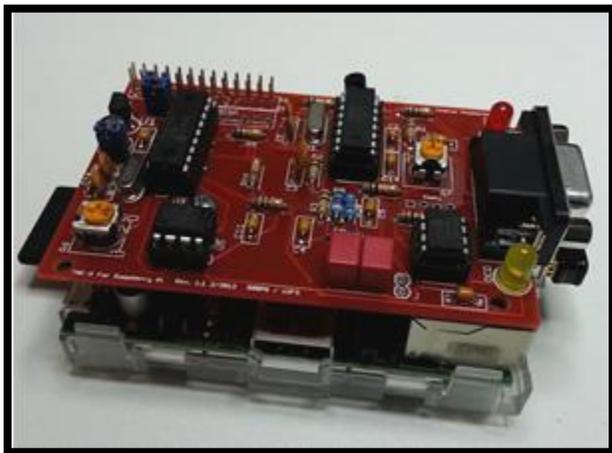


Figure 2: TNC-Pi mounted on a Raspberry Pi

Assembly Instructions

Parts List

Your kit should come with the parts listed in the table below. You can use the two columns of checkboxes to track your progress. As you inventory your parts, put a checkmark in the first column. Once you've installed a part, put a checkmark in the second column.

Inventoried	Installed	Designation	Description	Notes
<input type="checkbox"/>	<input type="checkbox"/>	C1	10 μ f electrolytic	Polarized
<input type="checkbox"/>	<input type="checkbox"/>	C2, C4, C5, C6, C11, C14, C15, C21	0.1 μ f monicap	
<input type="checkbox"/>	<input type="checkbox"/>	C7, C8	0.01 μ f 2.5%	Red and yellow or grey; if grey marked 10 nF
<input type="checkbox"/>	<input type="checkbox"/>	C9, C10	18 pf ceramic disk	
<input type="checkbox"/>	<input type="checkbox"/>	C12, C13	22 pf ceramic disk	
<input type="checkbox"/>	<input type="checkbox"/>	C3	100 pf ceramic disk	
<input type="checkbox"/>	<input type="checkbox"/>			
<input type="checkbox"/>	<input type="checkbox"/>	R1, R2, R3	100K resistor	Brown, black, yellow
<input type="checkbox"/>	<input type="checkbox"/>	R4, R17, R19,	1K resistor	Brown, black, red
<input type="checkbox"/>	<input type="checkbox"/>	R5, R11, R14, R16, R18	10K resistor	Brown, black, orange
<input type="checkbox"/>	<input type="checkbox"/>	R8	24.9K resistor	Red, yellow, white, red, brown
<input type="checkbox"/>	<input type="checkbox"/>	R9	9.31K resistor	White, orange, brown, brown, brown
<input type="checkbox"/>	<input type="checkbox"/>	R10	18.7K resistor	Brown, grey, purple, red, brown

<input type="checkbox"/>	<input type="checkbox"/>	R6, R7	10K trimmer potentiometer	Orange
<input type="checkbox"/>	<input type="checkbox"/>	X1	3.57 MHz crystal	
<input type="checkbox"/>	<input type="checkbox"/>	X2	20.00 MHz crystal	
<input type="checkbox"/>	<input type="checkbox"/>	D4	Red LED (PTT)	Polarized
<input type="checkbox"/>	<input type="checkbox"/>	D5	Yellow LED (DCD)	Polarized
<input type="checkbox"/>	<input type="checkbox"/>	Q1	PN2222 transistor	3 pin, flat side (don't confuse with U1)
<input type="checkbox"/>	<input type="checkbox"/>	U1	MCP1700-33 or MCP1700-30 Regulator	3 pin, flat side
<input type="checkbox"/>	<input type="checkbox"/>	U2	CML MX-614 Modem	16 pin IC
<input type="checkbox"/>	<input type="checkbox"/>	U3	PIC16F1847 microcontroller	18 pin IC
<input type="checkbox"/>	<input type="checkbox"/>	U4	MCP6023 Op Amp	8 pin IC
<input type="checkbox"/>	<input type="checkbox"/>	U5	23K640 Memory	8 pin IC
<input type="checkbox"/>	<input type="checkbox"/>	IC Sockets	For U2, U3, U4, U5	One 16-pin, one 18-pin and two 8-pin sockets
<input type="checkbox"/>	<input type="checkbox"/>	JP3, JP4	2-pin header	Combined into one 2x2 pin header
<input type="checkbox"/>	<input type="checkbox"/>		2 x 13 extra long header	
<input type="checkbox"/>	<input type="checkbox"/>		9 Pin D-Sub connector	
<input type="checkbox"/>	<input type="checkbox"/>		Printed Circuit Board	
<input type="checkbox"/>	<input type="checkbox"/>		2 shorting jumpers	

Assembly

Note: John McDonough has published a number of high quality photos of various stages in the assembly process. See <http://www.gsl.net/wb8rcr/BuildingTheTNC-Pi.html>

- Start by installing the parts that lie flat on the board. This includes:
 - a.** all of the 0.1uf monocraps and
 - b.** all of the resistors except for R6 and R7.

R10 and R11 have pads that are rather close together. Be careful that you do not accidentally create a solder bridge between them.

- Next install the two crystals.
Ensure that the 20 MHz crystal is the one nearest the 18 pin IC.
- Next install the IC sockets.
Ensure that the notch on the socket lines up with the notch on the IC outline on the PC board.

Do not plug the chips into the sockets at this point.

- Next install the rest of the capacitors.
With C1, make sure that the longer lead is placed in the hole marked with a +.
- Next install the two potentiometers (R6 and R7).
- Next install the two LEDs.
Ensure the shorter leads on the LEDs go through the holes closest to the flat side of the LED outline.

The LEDs can be installed with bent legs so the LEDs point toward the front of the board. This makes them easier to see when TNC-Pis are stacked one on top of the other. See Figure 1: Two TNC-Pi's stacked on one Raspberry Pi.

- Install the transistor.
Ensure you are installing the transistor rather than the voltage regulator... they look a lot alike. The transistor is not well marked on the board... it goes in between resistors R18 and R19.
- Now install the voltage regulator.
It is not well marked on the board either. It goes in the three holes above C1. Install it so that the flat side of the regulator faces away from U3.

If you have a board that is earlier than rev. 1.2, you will also find a 1x2 header that should be installed at JP7. More recent boards no longer have this part.

Now install the 9 pin D-Sub connector. Ensure that you push it all the way in so that it is flush against the board. In addition to soldering the pins, you'll gain mechanical stability by soldering the pins that go into the large round holes on the sides of the connector.

Note: The D-Sub connector is optional. You can install the Radio header to connect your radio to the TNC-Pi instead. See the section, [Connecting the Radio](#), for more information.

Solder in the 2 x13 header. This part is somewhat tricky.

- a.** Start by installing this jumper on your Raspberry Pi and then
- b.** lower the TNC-Pi board onto the connector so that the body of the connector is on the **BOTTOM** of the TNC-Pi board.
- c.** Now you'll need to solder the board about a half millimeter from all the way down in order to prevent the board from bumping into the USB connector on the Pi.

Your best bet is to carefully trim the solder leads on the part of the TNC-Pi board that will be on the USB connector side of the Pi to get as much clearance as possible. You might also want to put a piece of insulating tape on top of the USB connector just to be on the safe side. You can then solder the pins that come through the top of the board.

These pins will allow you to stack a second TNC-Pi on top of the first one if you choose to do so.

Now install the 2 x 2 jumper (JP3 and JP4).

Except for the very earliest kits, I've included 2 4-40 screws and a 5/8" spacer. If you have one of the new Raspberry Pi Rev. B boards, you can use this to provide some additional mechanical stability by putting it between the hole in the Pi and the hole in the TNC-Pi.

This is a work in progress though. The holes are just slightly misaligned, so it would be best to not tighten the screw on the Pi side until the screw on the TNC side has been partially installed.

If you are stacking 2 or more TNC's you might also find it preferable to use a spacer that is a male to female, rather than female to female. These male to female spacers are available on the TNC-Pi website.

- If you have a TNC-Pi board that is earlier than rev 1.2, install the jumper at JP7.
- Regardless of which board you have install jumpers at JP4 and JP3. They should be installed so they are parallel to the body of the 2x13 pin connector.

Leave the board connected to the Raspberry Pi and power up the Pi. Check the voltage between pin 5 (negative) and pin 14 (positive) on U3. It should read about 3.3 volts. With the notch at the top of the chip, these pins are the ones half way down the left side (negative) and right side (positive) of the chip.

If the voltage check is not successful, find and fix the fault before proceeding.

- Power down the Pi and remove the TNC-Pi board from the Pi, then install the 4 ICs.

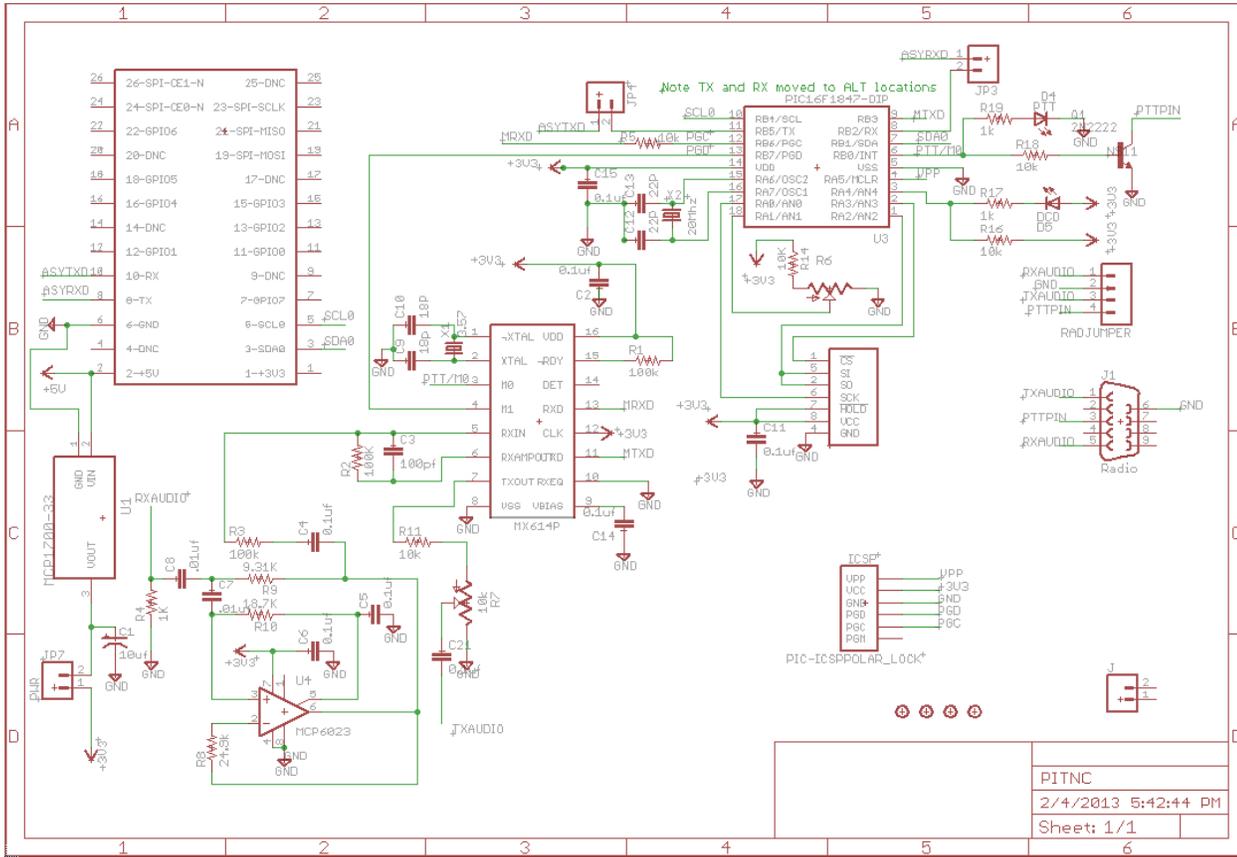
You may wish to bend the pins of the four ICs slightly inward to facilitate inserting them into their sockets.

Ensure the notch on the top of each chip lines up with the notch printed on the PC board. Also make absolutely certain that you do not mix up the two 8 pin chips and plug them into the wrong sockets.

Note: Nothing will be installed at the 6 pins marked ICSP or the pins marked J.

Congratulations, you're done assembling the TNC-Pi.

Schematic Diagram



(see next page for parts layout diagram)

Connecting the Radio

You can either wire up a 9 pin D-Sub plug to mate with the one on the TNC-Pi, or, if you'd prefer, you can use the four holes below `R7` labeled `Radio` to hard wire a radio connection. (No header is provided in the kit for this.)

If you use the 9 pin D-Sub connection, it should be wired as follows:

Pin 1: TX Audio
Pin 3: Push to Talk (PTT)
Pin 5: Receive Audio
Pin 6: Ground

Adjust transmit audio output

Potentiometer `R7` adjusts the level of the transmit audio output. Most people will find that they need to set it fairly near the minimum setting.

One way to set this is to use two radios, one to monitor the transmitted signal and the other connected to the TNC-Pi.

1. Key the radio connected to the TNC-Pi manually by pushing the PTT button on it.
2. On the other radio you will hear a continuous tone (even though no data is being transmitted... you don't need to be running any software on the Raspberry Pi to do this).
3. Turn `R7` all the way down and the tone will go away.
4. Then slowly turn it up until the volume doesn't increase any further in the monitor receiver.
5. When you reach this point, back it off just a little and you should have it about right.

Set the transmit delay

The transmit delay (`TXDelay`) can be set either in software or using `R6`. The default is to set it in software. To change the `TXDelay` parameter, you'll need to use the `pitnc_setparams` program as explained below in the section, Using Xastir.

If you set the value of `TXDelay` to 0 using the `pitnc_setparams` program, you can then use `R6` to set the `TXDelay`.

Configuring the Raspberry Pi

(Most of the information in the next two sections was provided by John Wiseman, G8BPQ.)

It might be a good idea to make sure your Pi operating system is up to date. You can do this from the command line by entering the following (this assumes you have either hard wired or wireless network connectivity):

```
sudo apt-get update
sudo apt-get upgrade
```

If you are going to use the serial port to connect the TNC-Pi to the Raspberry Pi, you will need to make a couple of changes to the Pi configuration. Edit the `cmdline.txt` file in the boot directory and make the following changes:

1. Remove the following: `console=ttyAMA0, 115200`
2. Remove the following: `kgdboc=ttyAMA0, 115200`

Note: `cmdline.txt` is a single line of text.

Then edit the `inittab` file in the `etc` directory and remove the line that says:

```
T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100
```

After removing these lines you will have to reboot the Pi so they can take effect. Unless you like to do command line editing, the easiest way to edit these file will be to start the raspberry Pi GUI interface. Then run LXTerminal and type the following:

```
sudo leafpad
```

This will run a very nice GUI editor with superuser privileges to allow you to edit system files.

Keyboard to Keyboard Connections

Note: For the following steps the Raspberry Pi must be connected to the internet.

You can use the Linux AX.25 routines to do connected mode packet. To do this you need to install the ax25 apps and tools. From the command line enter the following:

```
sudo apt-get install ax25-tools
sudo apt-get install ax25-apps
```

You'll need to configure it for your callsign by editing the `/etc/ax25/axports` file. You'll find two lines that allow you to enter your callsign. I did so as follows:

```
1      W2FS-1  19200  236      2          TNC 1
2      W2FS-2  19200  236      2          TNC 2
```

The serial rate must be at 19200, because that is the only baud rate support by TNC-Pi. The next two values (236 and 2 in the above example) are the values for `pacLen` and `maxframe`. Don't leave any blank lines in this file.

Now attach the serial port to the AX.25 system using `kissattach`:

```
sudo kissattach /dev/ttyAMA0 1 10.1.1.1
```

The number in blue above matches the port number from the `axports` file above. The number in red is an IP address. It's required here even though you aren't using the IP protocol on it.

Note: If you are using the IP protocol on your TNC-Pi your address should conform to the local convention for IP routing. The address in that case will almost certainly start with 44.

You can monitor packets on this channel by entering:

```
sudo axlisten -a
```

You can connect to another station for keyboard to keyboard QSO's as follows:

```
axcall 1 hiscall
```

where `hiscall` is the call station of the station to which you want to connect.

There seems to be a bug in the `axcall` routine. The first time you use it after calling `kissattach`, it will take significantly longer for the Pi to send a valid connect string to the TNC than it does in subsequent attempts. You may have to wait 10 – 15 seconds. Further attempts occur instantaneously. You can abort this first try by issuing a `Ctrl-C` and then issuing the `axcall` command again. It will then connect immediately.

Support

If you have any questions about your TNC-Pi or are having hardware issues with it, please contact John Hansen, W2FS at john@coastalchip.com. Software issues, particularly with regard to the Linux version of BPQ are best addressed to the author, John Wiseman, G8BPQ.

Additional information can be found on the TNC-Pi website: <http://tnc-x.com/TNCPi.htm>

The configuration programs `setparams` and `getparams` are available on the website as well: www.tnc-x.com/params.zip

Application Notes

Using Xastir

Thanks to M0HPJ for the following information

Another popular packet program that works well on the Pi is the APRS program Xastir.

Note: For the following steps the Raspberry Pi must be connected to the internet.

Update the package list

To install Xastir from the command line, first update the package list as follows:

```
sudo apt-get update
```

Then install Xastir

```
sudo apt-get install xastir
```

Kill `kissattach`

If you have been using the AX25 apps, you'll need to unload `kissattach` before Xastir will work. You can do this by simply rebooting the Pi, or you can kill the `kissattach` process from the command line. To do the latter, first find out what process number it is by using a `ps` command to list running processes:

```
ps -A|grep kissattach
```

Inspect the list to see the process number of the `kissattach` process, then kill the process using the `kill` command. For example, if the process number is 2335, issue the command:

```
sudo kill 2335
```

At this point it may be a good idea to rerun the `ps` command to be certain that the `kissattach` process was successfully killed.

Note: If you have not been using the AX25 apps, it is still necessary to edit the `cmdline.txt` and `inittab` files as described in the section on configuring the Pi.

Start Xastir

Then start the X-Windows GUI system with the following command:

```
startx
```

Xastir can be found in the programs menu under "other". The first time you run Xastir you'll be asked to put in your station parameters. Then you'll need to specify the port that the TNC-Pi is on. Click on Interface and then Interface Control. Click "Add" and pick "Serial KISS TNC" off the list. Push Add and it will bring up a properties list. Under TNC Port enter:

```
/dev/ttyAMA0
```

Running Xastir

Everything else is pretty self explanatory. Do make sure you change the baud rate to 19200. This is not the default. You can select "Maps" and the Map Chooser to select a better set of maps than the default. If you currently have internet connectivity, one of the cloud map options is probably best. If you select a new map option, you should deselect the old one. I haven't yet explored the option of

caching the maps to the local SD card so that I don't need an Internet connection, but I expect to look into that soon.

More information

A much more detailed discussion of running Xastir on the Pi in a mobile environment is contained in a paper presented at the 2013 ARRL/TAPR Digital Communications Conference. It can be found at:

<http://www.tnc-x.com/DCC2013.doc>.

Configuring TNC-Pi for Use with the I2C Protocol

The TNC-Pi can be configured to communicate with the Raspberry Pi using the I2C protocol. To configure the TNC-Pi for I2C, you will need a configuration programs `pi_tncsetparams` and `pi_tncgetparams`. The `getparams` program reads the parameters from the TNC-Pi while the `setparams` program allows you to set them. These programs are available at:

www.tnc-x.com/params.zip

Setup

Before using them, however, it will be necessary to make some additional configuration changes to the Pi. First, make the following changes to the following files:

1. In the `/etc/modprobe.d/raspi-blacklist.conf` file, remove the line: `blacklist I2C-bcm2708`
2. In the `etc/modules` file, add the line: `I2C-dev`

Get and set parameters

Note: Before running the `getparams` and `setparams` programs, ensure that `kissattach` is not running.

The `pitnc_getparams` program lists the values for all of the user-settable parameters. Its syntax is:

```
pitnc_getparams b d
```

where:

`b` is the number of the I2C bus and

d is the number of the I2C device on that bus.

TNC-Pi is shipped with both of these parameters set to 0. An I2C device number of 0 means the TNC is using the serial port rather than the I2C port.

The `pitnc_setparams` program writes values to the TNC-Pi. Its syntax is:

```
pitnc_setparams b d p v
```

where

once again b and d are the bus and device,

p is the parameter number (1 – 7) and

v is the new parameter value.

The parameters that you are most like to use are:

Parameter value	Description	Notes
1	Sets the TXDelay	With TNC-Pi, you can set the TXDelay either in software using the <code>pitnc_setparams</code> program or in hardware using R6. Setting this parameter to 0 causes the TNC-Pi R6 potentiometer to determine the value.
7	Sets the I2C address	Setting this to 0 causes data communication on the serial port (default), any other value up to 63 will cause I2C to be used with the TNC using this address. This permits you to use multiple TNC's with the same Raspberry Pi.

The BPQ software for Raspberry Pi is currently in beta and can be found at:

<http://www.tnc-x.com/InstallingLINBPQ.htm>

Running Applications other than LinBPQ with TNC-Pi in I2C mode.

John Wiseman, G8BPQ, sends along the following information about running other applications using I2C:

The TNC-Pi can be used with applications that use the Linux ax.25 stack, or applications that expect to see a KISS TNC on a serial port. Program `I2ckiss`

converts the I2C protocol to a standard KISS presentation on a virtual serial (pty) port. It is available here:

<http://www.tnc-x.com/I2ckiss.zip>

One copy is run for each TNC-Pi. The first two parameters to I2ckiss are I2C bus, I2C device. If using the kernel ax.25 code, then specify the port number (from axports) and the ip address. To use with other software, specify symlink and a symbolic name - I suggest com1 - com255

For example, to use with the Linux ax.25 stack:

```
sudo ./i2ckiss 0 16 1 10.1.1.1
```

i2ckiss will create a pty pair, and execute kissattach on the slave half, using the 3rd and 4th parameters

To use with a KISS application

```
./i2ckiss 0 16 symlink com1
```

i2ckiss will create a pty pair, then create a symlink to com1. The application would then be configured to use port com1.

Note: On Version 1 Pi boards (without mounting holes) the I2C bus number is zero, for the Version 2 boards it is 1.

Steps to create an APRS receive-only igate using the Raspberry Pi and TNC-Pi

by Paul Fischer, KC9RGZ

Equipment needed

- a) Raspberry Pi model B
- b) TNC-Pi (from Coastal ChipWorks at <http://tnc-x.com>)
- c) Power Supply
- d) Blank SD card (4 GB?)
- e) Windows PC to format and load the SD card
- f) Ethernet cable and router/switch with access to the Internet
- g) USB Keyboard
- h) Monitor (I use an old analog TV with RCA jack input)
- i) USB Mouse – only if you want to run the graphics environment on the Pi. Not required for text only setup.

j) 2M radio or scanner set to 144.39MHz (at least that's the frequency here in the USA!)

Instructions

1. Download the boot image for Raspbian "wheezy" (2013-05-25-wheezy-raspbian.zip) from <http://www.raspberrypi.org/downloads> and extract the image (.img) file.
2. On a Windows PC, format the SD card using **SDFormatter4exe.zip** – also available from <http://www.raspberrypi.org/downloads>. Use the options Format Type "QUICK" and Format Size Adjustment "ON".
3. On a Windows PC, use **Win32DiskImager** to write the image on an SD card (also downloadable from above site). Click the folder icon to the right of the Image file box to select the image file from step #1. Use the pull down tab to select the SD card and hit the "Write" button. This process takes a few minutes.
4. Put the SD card in the Rpi; connect the network cable, monitor and keyboard. Then the power supply and it should boot into the **raspi-config** application. This application can be run again once logged in by running "sudo raspi-config").
5. In the `raspi-config` program set these things:
 - a. Internationalisation Options; `Change_locale` – choose `"en_US. UTF-8 UTF-8"`
 - b. Internationalisation Options; `Change_timezone` – America; Central
 - c. Internationalisation Options; `Configure_keyboard` – set to `"Generic 105-key (Intl) PC; Other; English (US) – English (US,alternative international)"`
 - d. Overscan (Advanced Settings) – set to `enable`
 - e. Hit right arrow key and select the "Finish" key
 - f. Reboot: `"sudo shutdown -r now"`
6. Login with login "pi" and password "raspberrry"
7. Upgrade to root privileges `"sudo su"`

8. If the monitor doesn't display well (mine didn't), edit the file `/boot/config.txt` and uncomment the line:

```
overscan_left=16
```

And then re-boot. The above line will move the display to the left 16 pixels. You can play with the other settings as well until you are happy with the display.

9. Verify the software is up to date (this could take a while)

```
apt-get update  
apt-get upgrade
```

10. Edit the file `/boot/cmdline.txt` and remove the following parts of the line:

```
console=ttyAMA0,115200 kgdboc=ttyAMA0, 115200
```

11. Edit the file `/etc/inittab` and remove the line that says:

```
T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100
```

12. Reboot the device using the command:

```
shutdown -r now
```

Then log back in using `userid` and `password` listed above and upgrade to root privileges again.

13. Get the aprx digi/igate software:

```
cd /usr/src  
wget http://ham.zmailier.org/oh2mqk/aprx/aprx-2.05.svn485.tar.gz  
tar xvf aprx-2.05.svn485.tar.gz  
cd aprx-2.05.svn485/  
./configure  
make  
make install  
mkdir /var/log/aprx
```

14. Save a copy of the original aprx configuration file

```
cp /etc/aprx.conf /etc/aprx.conf.orig
```

15. Edit the file `/etc/aprx.conf` and make sure the following lines are included (please use your own call sign and latitude/longitude). Many of these lines are

already included, others just need to be uncommented. Just make sure all of these are included:

```
mycall KC9RGZ-11
<aprsis>
  server noam.aprs2.net
</aprsis>
<logging>
  pidfile /var/run/aprx.pid
  rflog /var/log/aprx/aprx-rf.log
  aprxlog /var/log/aprx/aprx.log
</logging>
<interface>
  serial-device /dev/ttyAMA0 19200 8n1 KISS
</interface>
<beacon>
  beaconmode aprsis
  cycle-size 10m
  beacon symbol "/&" lat "4124.57N" lon "09041.26W" comment "RasPi
  Rx-only iGate"
</beacon>
```

16. Verify the serial port is available:

```
chmod 666 /dev/ttyAMA0
```

17. Add the following lines to the end of the /etc/rc.local file to start the aprx program:

```
printf "Start the igate daemon... \n"
aprx
```

18. Reboot

```
shutdown -r now
```

19. Log back in and verify the processes are running

```
ps aux|grep aprx
```

20. Connect the radio and ensure you are gating data to the aprs network (check the files in /var/log/aprx)

21. Enjoy!

Notes:

1. Before the final move to "production", you should change the password of the "pi" userid!
2. To backup the SD card (after you have it all working, so you don't have to run the above instructions again!) – On a Windows PC, run the program HDDRAWCopy1.02Portable.exe (available from <http://hddguru.com/software/HDD-Raw-Copy-Tool/>). Select the SD card as the SOURCE and hit Continue. Double Click the "File" line to specify the name and location of the backup file (ex. 2013-07-03-wheezy-raspbian-backup). Use a file type of "Raw image (dd image) (*.img)". Hit Continue, then Start.
3. Restore is done by reversing the source and destination (use the file as the source and the SD card as the destination).