Raspberry Pi Applications in Digital Communications: A Mobile Xastir-Based APRS Station

John A. Hansen, W2FS
Department of Computer Information Sciences
State University of New York at Fredonia
john@hansen.net

Abstract

TNC-Pi is a version of the TNC-X KISS mode TNC that is designed specifically to plug into a Raspberry Pi computer. Coupled with a mobile monitor, and keyboard/trackpad, it makes it possible to assemble mobile APRS station for a very low cost. This paper describes the components needed for this and provides step by step instructions on how to configure it.

Introduction

The Raspberry Pi is a credit card sized computer based on a proprietary Broadcom processor that includes a 700 MHz ARM11 processor. The 'B' Model of the Pi has 512 MB of on board RAM and 2 USB ports as well as on board video, audio and Ethernet adapters. It's "hard drive" is any standard (4 GB or larger) SD card, which plugs into the on board SD card socket. You power it with 5 volts, typically from a standard cell phone charger. The 'B' model costs $35. The somewhat less powerful 'A' model (256MB of RAM, 1 USB port, no Ethernet) costs $25. It is designed to run Linux, most commonly a variant of Debian that is called "Wheezy". The standard Wheezy distribution contains both a command line and a GUI user interface.

The Raspberry Pi was developed by the Raspberry Pi Foundation, a UK charity, whose goal was to provide school children with an inexpensive platform to learn computer programming. It is called "Raspberry Pi" because there is a long history of naming computers after fruit (Apricot, Apple, Tangerine) and because the educational programming language of choice for it is Python (hence, Pi).

While it was originally designed to be an education tool for children, it was immediately latched onto by the maker community in general and the amateur radio community in particular. It is being used for a wide range of communications applications in ham radio including:

- As a stand-alone IGate.
- As a RMS server for WinLink
- As a WinLink client
- With a GMSK board to provide a stand alone D-Star Hotspot.
- For keyboard to keyboard packet communication.

- Doing HF digital communication, typically using Fldigi.
- For APRS, most often using the Linux program Xastir.

Moving beyond purely digital applications, other amateurs are putting the Pi to even more esoteric applications. Guido Dolle PE1NNZ, for example, has developed methods to use the Pi as an actual radio transmitter, include digital WSPR, FM and SSB modes.

This paper will focus on assembling a mobile system for running the graphical Xastir APRS package. It will include step by step directions on getting this up and running, with the target use case being a mobile emcomm communications platform. The design specifications are:

1. It must run on an automobile 13v supply.
2. It must provide a visual map presentation of stations in the area.
3. It must support messaging as well as location information and beacons.
4. It must NOT rely on any Internet connection, even for maps.
5. It must be inexpensive.

Needed Hardware

This project requires a 2 meter transceiver, a TNC, a Raspberry Pi, a monitor, keyboard/trackpad, and a power supply. Optionally, you might also want to add a GPS receiver if you want real time updates of your position to be transmitted. The transceiver can be virtually any 2 meter unit, mobile or HT as long as it is suitable for operating from a mobile environment. For a TNC, there are several options. The Raspberry Pi has serial port TX and RX lines on it's expansion header, but they are at 3 volt levels, not RS-232 levels. So it would be possible to use a conventional TNC if you used an adapter to make this conversion. Some individuals have reported problems using serial to USB converters (like the one in TNC-X) with the Pi's USB ports, so this approach is probably not ideal. The easiest way to add a TNC is with a TNC-Pi, which is a plug-in 1200 baud TNC designed specifically for the Raspberry Pi header. It mounts directly on top of the Pi and is held in place by a fixed standoff. (See the TNC-Pi section for details on this TNC).

There are a couple of choices for a monitor. I happened to have a 12 inch 12 volt monitor with a VGA output. I plugged the VGA output into a VGA to HDMI adapter and plugged this into the Raspberry Pi. It worked very well. Probably a more appropriate monitor for a mobile platform would be a monitor of the type designed to either show DVDs or the image from a backup camera. Both of these types of displays run on 12 volts and have standard RCA video outputs that can plug into the Raspberry Pi.

A wireless keyboard/trackpad combination is probably the idea console for this mobile platform because you won't have any additional wires running around the car. The easiest way to provide 5 volts for the Pi is to use a cigarette lighter adapter that provides USB charging outputs. It is not difficult to find one of these that provides an amp or two of current (the two amp ones are primarily designed to charge iPads). Since the Pi draws less than 700 ma (even when powering other devices via its USB ports) and the TNC-Pi draws less than 30 ma, the lighter adapter should be more than adequate. I'd look for one that has two USB outlets as

well as a cigarette lighter jack, which could be used for powering the monitor. BUD makes an enclosure called the "Pi Sandwich" that holds both the Raspberry Pi and the TNC-Pi board. It is ideal for this project, unless you plan to stack more than on TNC on a single Pi. Here's a summary of my budget for the project (not including the 2 meter radio):

1. Raspberry Pi	$35
2. 8 GB SD card	$10
3. TNC-Pi kit	$40
4. 7" LCD monitor	$32
5. Logitech Wireless Kbd/Trkpad K400r	$30
6. Auto Lighter USB Adapter	$15
7. BUD PS-11591 Pi Sandwich Enc.	$6
Total Cost	$168

There is an issue with the K400r. I noticed in on-line reviews some users said it worked great and others said that they could not get it to work with the Raspberry Pi. I had both kinds of experiences. When using it with a the 7" LCD monitor with RCA video output, it worked great. When I tried to use with a VGA monitor and HDMI to VGA adapter, it did not work.

Though technically not necessary for this project, you might find a couple of other items useful. Since you are almost certainly going to configure this setup in your home, not your car, you would find it useful to have a 5 volt power supply that plugs into an AC socket. Use almost any old cell phone charger for this purpose. You will need some type of network connection to allow the Pi to update it's software and load Xastir. If you have a wired network, you can just plug in an Ethernet cable, but if, like most people, you rely entirely on wireless networking, you'll need a USB Ethernet adapter. Edimax makes a very nice one that is widely available for $10-12. If you want to use one of your VGA monitors with the Pi, you'll also need a VGA to HDMI adapter to make it go (~$10). You also might want to consider a USB hub to allow you to connect more than two USB devices at the same time.

TNC-Pi Overview

TNC-X is a kiss mode only TNC that has been available for over 10 years. When I originally designed it, I made the decision to make the firmware open source. The theory behind open source is that by making the code widely available, others will come along and suggest improvements or new applications for it. In the first 10 years that TNC-X was available this simply didn't happen. Then last year I was approached by John Wiseman, G8BPQ. If that callsign sounds familiar it may be that you remember it from the earlier years of packet radio when John designed the very widely used BPQ packet networking software. Most of the people running bulletin boards in the 1990s were also using John's networking software.

John took the TNC-X source code and modified it to support the Inter-Integrated Circuit (IIC or I2C) protocol. The I2C protocol allows one to hang a bunch of different devices on a single bus, and give them individual addresses so they can all communicate with a single host without the need for separate connections for each of them. John also made some hardware changes to allow the revised TNC-X to plug directly into a Raspberry Pi. Even better, with the I2C protocol, it is possible to stack multiple TNCs onto the same Raspberry Pi. John then

modified the BPQ networking software so that it supports the I2C protocol. In addition to supporting I2C, this new device (called TNC-Pi) still supports a serial connection to the Pi as well, so it can be used either way. The new version is available through Coastal ChipWorks (www.tnc-x.com) just as the original TNC-X is. What's more, because there are fewer parts in this new version, it was possible to lower the price of the device by 20%.

Taking Your First Byte of Pi

As mentioned above, the easiest way to get your mobile APRS station up and running is to do so in a more permanent location that has full time internet access. The Raspberry Pi is designed to boot off an SD card. To do this, you need to put the disk image for the Wheezy Debian distro on an SD card and then place it in the Pi's SD card slot. The first step is to download the image from the Raspberry Pi downloads page:

http://www.raspberrypi.org/downloads

For this project, pick the Raspian Wheezy distribution, not NOOBS. Obviously, you won't be downloading this image using your Pi, since your Pi has nothing to boot from yet. Make the download onto your regular PC. The size of this download is almost 500 MB, so depending on the speed of your network connection, it may take a while. The file is a zip file so you will have to unzip it before you can use it. Once you unzip it you will have a .img file that you will use to create the boot disk for the Pi. This paper assumes that you are doing this on Windows machine. If you are doing it on a Linux or MacOs machine, see the instructions here:

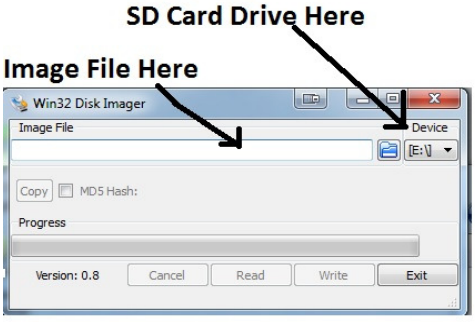
http://www.youtube.com/watch?v=DX8uqyEpmd0

If you are doing this on a Windows machine, first get the free software Win32 Disk Imager from:

http://sourceforge.net/projects/win32diskimager/

If your computer has an SD card reader in it, put your SD card in the reader. If it doesn't have a reader, you'll need to get one that can plug into a USB port. You could put this image on an SD card as small as 4 GB, but 8 GB cards are only slightly more expensive and you will probably want considerable room on the card to store map files after you get Xastir running.

Run the Disk Imager program. At the top of the program window on the left you'll see a box where you specify the image file (the folder next to it will allow you to browse for it) and on the right you select the drive on your computer that is the SD card. Be absolutely certain that you pick the drive letter that is the SD card. If you have a USB hard drive it will show up in this list as well... it would be a very, very bad thing if you selected it instead of the SD card! Then all you have to do is press the "write" button and wait. This will take a while, but you can see your progress on the progress bar.



The image is only about 500 MB and you are putting it on a multi-GB SD card. Initially you won't have the full capacity of the SD card available to you. Don't worry about this, we'll expand it in a little bit.

Now you are ready to boot your Pi for the first time. Insert the SD card (note it will go in upside down). Connect your keyboard/trackpad USB dongle and your monitor. You will have an easier time if you start with a full size monitor, not the 7 inch one. Either plug in a network cable or insert your Wifi USB dongle. Now plug in the cell phone charger and power up the Pi. After a few seconds you should see a lot of stuff scrolling by on the screen. The user code for the Pi is:

Pi

and the password is:

Raspberry

You can change both of these later if you want to. Initially you will boot into a screen that will help you configure the Pi. There are just a couple of items here you will want to do. There is an option to configure the keyboard; you'll probably want to take advantage of that, especially if you are in the US. There is also an option to expand the image size to the full size of the SD card (expand_rootfs). This will reclaim the rest of the available space on the card for you, but it will not take effect until you reboot the Pi. There is also an option to allow you to boot directly into the graphical user interface (GUI) instead of the command line. If you do this, you will not have to put in your usercode/password every time you boot up. It will still be possible to access the command line from the GUI, so you don't lose much by enabling this. To exit the configuration program tab down to finish and hit return. If at some point in the future you want rerun this configuration program you may do so by typing:

sudo raspi-config

at the command line.

A note on sudo: If you're not familiar with Linux you'll probably wonder about all the Linux commands that appear to start with "sudo". Sudo allows users to run programs that they would not normally have the privileges necessary to run. In the case of the examples you are seeing here, these programs that must be run by a superuser.

Now reboot the system by typing the following at the command prompt:

sudo shutdown -r now

This will cause Linux to start the shutdown sequence. The -r specifies that you want it to reboot. If you wanted it to shutdown and not reboot, you should use -h instead.

When the system reloads you are going to need the network to be available. If you did not opt to boot directly to the GUI, you can start the GUI by typing:

startx

at the command prompt. If you are using Wifi, you can configure your Wifi connection by double clicking on the WiFi Config icon. When the WiFi Config screen comes up, push the "scan" button to get a list of available networks, select yours, and push the connect button to connect to it.

Next you will need to edit some system files. Double click on the LXTerminal icon on the desktop (which will open a command line) and type:

sudo leafpad

at the command line prompt. Leaf pad is a nice GUI editor and you must run it using sudo to have adequate privileges to edit system files. Start by clicking file/open and select "File System" on the left. Then double click on "boot" and double click on cmdline.txt to open that file. You'll find that it is a file that contains just one line of text. Look for the part of that line that says:

console=ttyAMA0, 115200

and remove that part. Then look for:

kgdboc=ttyAMA0, 115200

and remove it as well. Then save the file.

Then run sudo leafpad again and this time open the etc directory instead of the boot directory. Scroll down until you find the file named inittab and double click on it to open it. Find the line that says:

0:23:respawn:/sbin/getty –L ttyAMA0 115200 vt100

and delete it. Then save the file. You'll need to reboot in order for these changes to take effect. You can do this right from the GUI by using LXTerminal and typing the shutdown command at the command line. Alternatively, if you booted into GUI interface, you can shutdown or reboot by clicking the red button in the bottom right corner of the screen.

The next step is to update the software and then install Xastir. You'll need to do this either from the command line or from the LXTerminal application in the GUI. You'll need to have an Internet connection to do this.

Start with:

sudo apt-get update

to update the list of available packages and:

sudo apt-get upgrade

to install the latest version of the software you already have.

Then you can install xastir by typing:

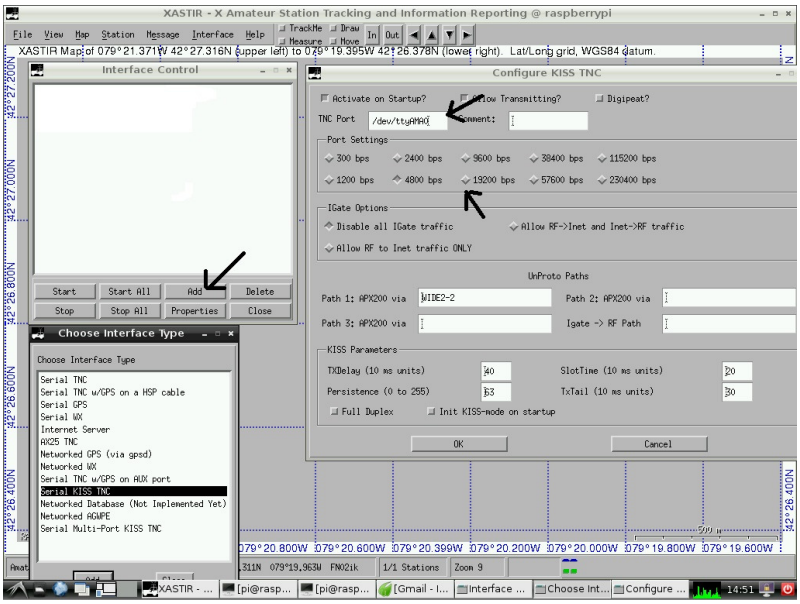
sudo apt-get install xastir

This will take a while, so this would be a good time for a coffee break.

When it finishes, if you have not already done so, this would be a good time to install the TNC-Pi on your Raspberry Pi. Refer to the on-line manual for TNC-Pi for the instructions on building the TNC and connecting it to your radio. Be careful to make sure that the TNC-Pi board does not short against the USB connector on the Pi. Note that the on-line manual also has instructions for using AX25 to do keyboard to keyboard packet, however if all you plan to do is run Xastir you can ignore that section.

Getting Started with Xastir

After the Pi reboots, you'll have to start the GUI to be able to run Xastir. Click on the icon in the bottom left corner of the screen (where the start button would be in Windows) and select xastir off the "Other" menu. You'll have to scroll down a bit to get to Xastir. 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. To do this 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:

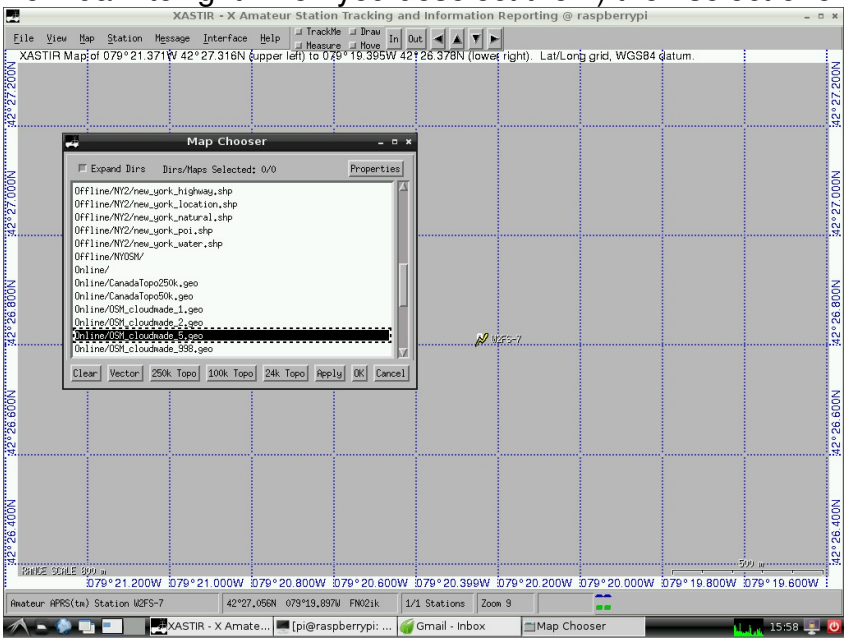


/dev/ttyAMA0

Also, make sure that you specify a baud rate of 19200, since this is the only baud rate TNC-Pi supports. You will need to make this change because this is not the default.

About Maps

Xastir is capable of using maps in quite a few different formats. All of these maps can be divided into two basic categories though: on-line and off-line. The easiest maps to use are on-line since the map information is downloaded over the web as you need it. In that case you don't actually have to load any maps onto your computer. However, you will find that on a Raspberry Pi this option is pretty slow (you will need patience!). In addition, this project is designed to work without an Internet connection. However, to get a feeling for how Xastir works, it might be a good idea to spend some time playing with the on-line maps before moving to the offline maps. To get started quickly with online maps, go to the map menu and pick Map Chooser. This will open a list of maps that are currently available. Before doing anything else, deselect any maps that might already be selected by clicking on them (they turn from dark to light when you deselect them) then select one of the online "cloudmade" selections (for example Online/OSM_cloudmade_5.geo) and click OK. It will take a little while to load.



When you are ready to start using offline maps you'll need to download them. I find it easier to download the maps on another computer, extract the needed data, copy it to a thumb drive and then plug the thumb drive into the Pi and copy the map files to the Pi. You may find that you prefer to download the map files directly onto the Pi. Either way, you first have to know where to get the maps.

While Xastir is capable of dealing with maps in a variety of formats (you can even make your own maps, if you like) the ones you are most likely to use are from OpenStreetMap. OpenStreetMap is a project that is designed to make worldwide map information freely available. The Tiger Map project from the US Census Bureau was incorporated into OpenStreetMaps in the mid 2000's but since then the maps have evolved as a result of the contributions of individuals. OpenStreetMap was the basis for the online CloudMade maps referenced above. In addition CloudMade maps can be downloaded and used offline. To get the maps for a particular area go to:

http://downloads.cloudmade.com/

Here you will drill down through a series of links to find the maps for the area that you are interested in. You don't need all the files; you just need the one that ends in shapefiles.zip. It is not practical to load up all the US Map data in one go. The file is available, but it would be too large to fit on even a

32 GB SD card. So when using this with the Raspberry Pi, it probably makes more sense to download the areas that you are interested in. In emcomm work, this is probably not much of a restriction since emergencies are generally regionally limited. It is practical to download multiple states to your Pi. Remember however, that the amount of time it takes to display a map is a function of how many maps you have selected, so you can speed things up by only selecting the maps that you actually need.

Loading maps into the Pi involves putting the files in a directory structure. I found the easiest way to do this on the Pi was to use the File Manager program from within the GUI. It can be found under the "Accessories" menu item. When you were using the Map Chooser on Xastir, you might have noticed that the map selections looked a lot like a directory structure. That's exactly what they are. Whatever directory structure you decide to put your map files in is how they will be presented in the Map Chooser. The maps are located in the Raspberry Pi as follows:

/usr/share/xastir/maps

So I created a subdirectory within the maps subdirectory called Offline (there's already one there called Online). Then I created a subdirectory within that called NY (for New York maps) and put the New York files I downloaded in there. The map files that were downloaded from CloudMade fall into seven categories:

administrative
coastline
highway
location
natural
poi
water

Highway is by far the largest of these and the most important. It contains not just highway information, but street information too. Within each of the seven categories there are four types of files. .dbf, .prj, .shp, .shx. Only the .shp files show up on the Map Chooser, but you need to put all of the files in this directory in order for the maps to work.

To use the downloaded maps, open the Map Chooser. You will see both the directory you created to hold these mapfiles and also individual listings for each of the seven .shp files. You can select all of the files in the directory by simply selecting the directory that contains them or you can select the files individually. The poi file contains information on "Points of Interest". At least in my area there are a LOT of these; so many that they clutter up the map. I find the map (at least in my area) to be a lot more useful if I deselect the poi file to turn them off.

Navigation and Other Features

The easiest way to zoom in on a map is hold down the left mouse button and then drag to draw a rectangle of the area you want to zoom in to. There are also zoom in and out buttons at the top of the screen. In addition, if you right click on the map, a menu will pop up that will allow you to zoom around the point where you clicked. Unlike many other applications, the

mouse wheel does not adjust the zoom level. Instead, the mouse wheel allows you to scroll north and south on the map. You can also use the cursor keys to move around or the arrow buttons on the top of the screen. You can move to have the map centered on a particular station by picking the item called "find station) from the station menu.

Because Xastir is a full featured APRS program, you can do all of the things with it that you would expect. I included a wireless keyboard/trackpad with my station design primarily for the purpose of doing APRS messaging. Xastir also has the capability to function as an iGate, though if that is the only thing you plan to use it for, there are simpler solutions that run on the Raspberry Pi that don't use as much of the Pi's resources.

Moving On to I2C

So far we have used Xastir entirely with the Pi's serial port connected to a single TNC. But what if you want have second TNC stacked on the Pi and use it to connect to a second radio on another channel? This could be for the purpose of running APRS on a second frequency or it could be for another application entirely. Suppose, for example, you are keeping track of the locations of a number of mobile stations using Xastir, but you also want to send and receive email using paclink on another frequency? Since the Pi only has one serial port it would be necessary to communicate with the TNCs using the I2C protocol instead. Xastir is not designed to utilize I2C to communicate with TNC's, but G8BPQ has written a small utility called "i2ckiss" which presents programs like Xastir with a standard serial port interface and then forwards that data to an I2C address. As a result, any program that is capable of communicating serial data in KISS mode can now be used with the TNC-Pi set up for I2C communication rather than serial communication. In theory you could stack over 60 TNC-Pi's on a single Raspberry Pi, but the Pi's relatively limited processing resources would make this impractical (not to mention difficult to balance on the Pi!).

To use I2C with Xastir, it is necessary to make a couple of changes to the Pi and a couple of changes to the TNC-Pi. Using a text editor (like leafpad) edit the /etc/modprobe.d/raspi-blacklist.conf file and remove the following line (it may already be commented out):

blacklist i2c-bcm2708

Then edit the /etc/modules file and add the following line:

i2c-dev

Now you will need to change the TNC-Pi so that it uses I2C rather than serial communication. Download the following files:

www.tnc-x.com/params.zip
www.tnc-x.com/i2ckiss.zip

Once again I found it easier to do this on my PC, unzip them, and then transfer the files to the Raspberry Pi using a thumb drive (just as described above in the maps section). I created a subdirectory on my Pi called tncpi and put the files in that directory. You can put them

wherever you want as long as you keep track of where they are. If you run the pitnc_getparams program using the parameters 0 0:

sudo ./pitnc_getparams 0 0

you will see the following output:

```
01 TXDelay - Zero means use ADC 40
02 Persistance                  63
03 Slottime (in 10 mS)          20
04 TXTail                       30
05 Full Duplex - Not used       0
06 Our Channel (Hex)            00
07 I2C Address (0 = async) Hex   0
   ADC Value                    41        (your value may differ)
```

The only two values you'll be interested in here are TXDelay and I2C address. TXDelay is the amount of time the TNC will wait between keying the transmitter and sending a packet. It can be adjusted here in software, or it can be adjusted using the R6 potentiometer on the TNC-Pi board. To set it using R6, set the value of TXDelay here to 0. Any other value set here will specify the time in 10's of milliseconds of delay to use (40 = 400 ms).

Parameter 7 is the I2C address. A value of zero means we are using the serial port. That's the value that you should initially see. To use I2C, use an address in the range of 1 to 62. To make this change, use the pitnc_setparams program. For example, to change the address to 3 issue the command:

sudo ./pitnc_setparams 0 0 7 3

The first zero here is the I2C bus number. When in serial mode this will be 0, but after you've changed to I2C it will depend on the vintage of your Raspberry Pi board. If you have an older Pi that doesn't have mounting holes on the board, the bus address will be zero. If you have one of the newer boards that does have mounting holes, the bus address will be 1. Changes to the I2C address will not take effect until the TNC and the Raspberry Pi are rebooted. The seven indicates that you want to modify the I2C address parameter. The 3 is the new value of the parameter. Note that you enter this last number as a decimal value, but it will show up as a hexadecimal value when you do a pitnc_getparameters command. So if you enter 16, for example, it will show up as 10.

Now you will need to reboot both the Raspberry Pi and the TNC-Pi and then remove the jumpers from JP3 and JP4 on the TNC-Pi board. **If you find that you have "bricked" your TNC after changing parameter 7, it is probably because you forgot to remove jumpers JP3 and JP4.** Note also that it is necessary to completely remove power from the TNC-Pi to get it to reboot. Simply rebooting the Raspberry Pi will not be sufficient.

After the Pi reboots issue the following command:

sudo usermod -a -G i2c pi

This will give the user pi access to the i2c bus. This command only has to be issued once, not every time you boot the board. In fact, everything that we have done up until now is a one time setup procedure.

To run Xastir using I2C, you will have to build a connection between I2C and a symbol name that Xastir can treat as a serial port. To do this use the i2ckiss program that you downloaded. Switch to the directory that it is in and issue the following command:

./i2ckiss 1 3 symlink com1

A couple of notes: First, if you have one of the older boards with no mounting holes the 1 in the above command should be a 0. Second instead of the 3 in the above line, fill in whatever number you chose to use for the I2C address (enter the address in decimal, not hexadecimal). Third, you can use any com value you want for the last parameter. Just remember what you entered because you will need enter it when you get to Xastir. Finally, if you had multiple TNC-Pi's stacked on your Raspberry Pi, you would need to run i2ckiss for each one.

Now enter the GUI and run Xastir. You will need to modify the entry that you made in the interface control screen to specify com1 instead of /dev/ttyAMA0. Note, all you need enter is com1, there is no directory reference needed. Instead of modifying the interface that you previously specified, you could simply add a new one. You can then use the start and stop buttons to turn on and off the interface(s) that you want to use.

Using a Smaller Monitor

It makes a lot of sense to use a larger computer monitor when setting up your Raspberry Pi/TNC-Pi Xastir system. If you have room in your vehicle, you could continue to use a larger monitor there as well. If you google "12v vga monitor" you will find quite a few available, though at a price point that is significantly higher than the $32 specified above. Another possibility is to buy a 12v TV with HDMI input. Many of these are available because they appeal to campers and truckers.

I purchased an FDL 7 inch TFT LCD car rear view camera monitor from Amazon. It has a plug that mates with the RCA video input, an RCA jack to provide a second video input (unused in this project) and three other wires: red, black and blue. It comes with virtually no documentation. The red and black wires are positive and negative power, and the blue wire is used to turn the second input to the monitor on when the car is put into reverse. I simply clipped the blue wire because I had no need for it. Through experimentation I found that while the monitor is rated at 12 volts, it will continue to run well down to about 9 volts. It draws about 250 ma at 12 volts. It's really a remarkably good deal for $32 (and free shipping). But given my experience with the 7" monitor, I don't think I would consider using anything smaller.

If you have video connections made to both the HDMI and RCA jacks on the Pi it will always choose the HDMI output and not send video out the RCA jack, so it is necessary to unplug your HDMI (or VGA) monitor to get the RCA output to work. In addition, if you simply unplug the HDMI monitor, plug in the 7" monitor and reboot, you'll probably be fairly disappointed with

the results. When you boot the computer you will see the usual lines of text fly by on the screen, but the font will be so small you will be unable to read it. You have to get the resolution down to a level that is appropriate for an inexpensive 7" monitor in order for this approach to work.

You can make changes to the video output by editing the config.txt file that resides in the /boot directory. Simply add the following two lines to this file (you'll find an example commented out already in the file):

framebuffer_width=xxx
framebuffer_height=yyy

where xxx and yyy are the width and height, respectively, of the image you want to be shown on the screen. You should experiment with different values to see which ones you prefer. The larger these values, the more you will see on your screen but the blurrier it will get. I found a pretty good compromise for running Xastir was a width of 640 pixels and a height of 420 pixels. You will not be able to read text from the command line very well at this resolution, but it is about right for running Xastir. However, to read the street names, you'll need to increase the size of their font. You can do this by selecting "Configure" from the "File" menu and then clicking on "Fonts". The font that controls the street names is labeled "Station Font". I used the font chooser to select the "Bookman" font and set the size to 14. That is not a value that is available on the pull-down menu, but you can type it in and it will show up correctly on the map.

If you find that there is a black border around the screen image, you can get rid of this by uncommenting the line in config.txt that says:

#disable_overscan=1

On my display this resulted in the image being partly off the screen. Since I was using Xastir, this wasn't a bad thing since the top window border is pretty useless in any case. If you prefer, you can turn overscan back on and adjust the overscan by uncommenting and adjusting the following lines of code:

#overscan_left=16
#overscan_right=16
#overscan_top=16
#overscan_bottom=16

Lowering the numbers (even into negative values) will cause the image to get larger, while raising them will cause the image to get smaller. You can also gain some space at the bottom of the screen by hiding the panel that contains a row of icons. Right click it and select "panel settings". Pick the "Advanced" tab and click on "Minimize panel when not in use". The panel will then only appear when you point your mouse at the bottom of the screen.

You can change the background color of the maps by picking "Configure" from the "Map" menu. I found the default color was pretty dreary and adjusted it to "pale green". While you

are on this configuration menu, you can make the received stations show up much better by picking "Station Text Style" from this menu and changing it to "text on black".

There is one other display possibility that you might consider. If you do an Internet search on "Raspberry Pi remote desktop iPad" you will find that some people have successfully used an iPad as a Raspberry Pi display wirelessly by using a remote desktop client. This will, of course, require a network connection, but it won't necessarily require a connection to the Internet. In theory you should be able to connect using WiFi that is set to "ad hoc" rather than "infrastructure". Note that I have not actually tried this myself yet.

Backup

Finally, after you have the maps loaded and all the settings just the way you want them, it might be a good idea to back up your SD card on your PC just in case something happens to it. You can do this using the Win32 Disk Imager program that you used to make the original image. Specify a file name that you want to back up to and specify the drive that the SD card is in. Then press "Read". The one problem you might encounter with this is that all SD cards do not have the same capacity. Not even all 8 GB cards have the same capacity. They all vary somewhat. So if you happen to try to restore this image to a card that is slightly smaller than the one you initially used, you will not be able to do so. The easiest way to get around this problem is to initially buy 2 SD cards (not one) and use the smaller of the two to create your initial Wheezy image. That way if you need to restore, you'll know you are restoring to a larger card.