Practical Implementations of Bluetooth in Microcontroller Circuits

John A. Hansen, Ph.D. W2FS
Department of Computer and Information Sciences
225 Fenton Hall
State University of New York at Fredonia
Fredonia, NY 14063
john@coastalchip.com

Abstract

Bluetooth has become the standard technology for external interfacing to laptop PCs and mobile devices. This paper describes a number of ways to implement serial data communication using the Roving Networks Bluetooth devices.

KeyWords : APRS, TNC-X, Bluetooth, Roving Networks

Introduction

When Harold "Bluetooth" Gormsson (son of Gorm the Old) ascended to the throne of Demark in 958, he could never have imagined how dominant his namesake technology would become over 1000 years later. But Bluetooth (the hardware, not the king) is built into virtually every mobile device these days, including phones, tablets and laptop computers. This paper will provide a roadmap for working with Bluetooth interfaces in amateur radio projects.

Once upon a time, computer connectivity meant serial and parallel ports. In those days it was easy to build devices that would connect to these ports. However, both of these types of ports were later supplanted by USB (Universal Serial Bus). USB presented a problem… it comes in two flavors: "master" and "slave" (except in Los Angeles County, California where these terms are banned). USB requires a master to control the USB bus. Other devices on the bus are typically slaves. It is not possible to connect a slave to another slave without there being a master on the bus. So while a mobile phone may have a USB port, it is almost certainly a slave port since it's designed to be plugged into a PC. The same is true for GPS receivers. While this is being addressed to some extent by the USB "on the go" specification, you still don't see many TNCs or trackers that accommodate connections to mobile phones or USB GPS receivers.

A second problem is that USB generally requires device dependent drivers. Every USB device that you plug into your computer requires a specific driver. For some items (like thumb drivers and mice) the driver is already build into the operating system, but for most USB devices that you acquire, the first step to using them is to load a driver.

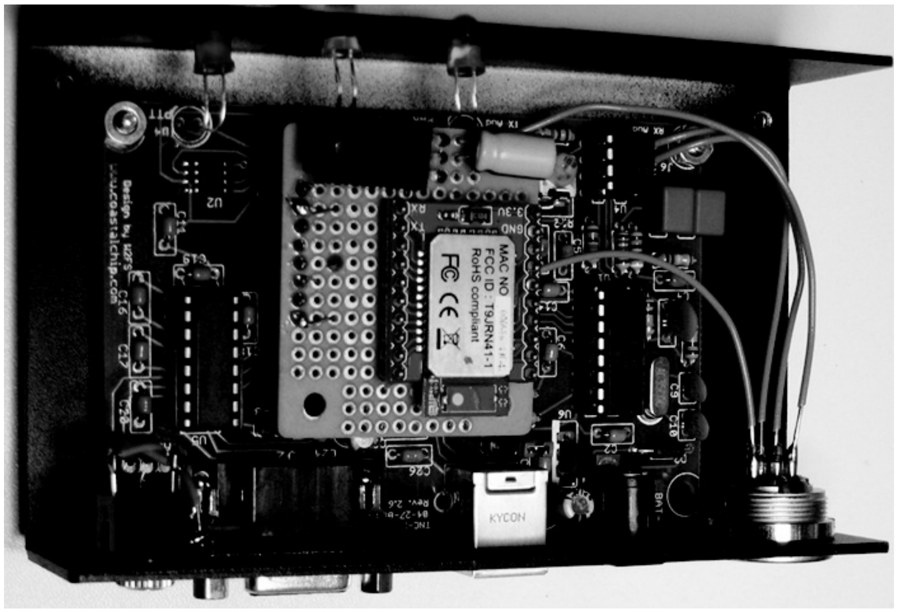
Just as King Bluetooth Gormsson united the warring factions in Scandinavia, Bluetooth the technology gets a wide range of devices to all work together. When you buy a Bluetooth headset it will work with both your PC (which is a USB master) and with your mobile phone (which is a USB slave).
In addition to getting past the master/slave issue, Bluetooth also has the advantage of being wireless. As a result it's a no brainer for experimenters to begin using Bluetooth for short range connectivity. With

my Bluetooth TNC I can connect to my DroidX phone running APRSDroid to display a map showing where all the local APRS stations are. There is no physical connection between the TNC and the Droid. The GPS in the phone provides the position information for my Bluetooth TNC to announce my position to the RF network. No serial cables needed; no USB cables needed.

Object-Oriented Hardware

The hottest thing in computer programming for the past couple of decades has been object-oriented programming. The reason that it is so important to software development is because it causes programmers to think of software as collections of components which can be mixed and matched in various combinations to achieve a range of functionality. Want to add a word processor to your program? No problem, just grab a word processor object and drop it in. You don't have to understand the internal programming or workings of the word processor; all you have to understand is the user interface.

The same thing has become true of hardware, but we don't call it "object-oriented hardware". Perhaps we should. Companies make components that can add functionality to your project and you don't have to know how they work; you merely have to understand how to interface them to your existing hardware. TNC-X was developed from the ground up using this concept. The X in TNC-X stands for expandability. The goal of this project was to develop a Bluetooth daughter board for the TNC-X so that it would be possible to connect it to Bluetooth enabled computer devices like my phone and my laptop computer. What I needed was an easy to use Bluetooth component that could be interfaced to TNC-X.



TNC-X with prototype Bluetooth daughter board.
In the center is the SparkFun board with the RN-42 Bluetooth module mounted. The antenna (blue) is just below the module (red).

Roving Networks Bluetooth Modules

The answer turned out to be found in Roving Networks Bluetooth radio modules (see Figure 1). These are relatively inexpensive Bluetooth and user-configurable. They come in two power levels: a low power version (Part # RN-42) which is 3 dBm and a high power version (Part #RN-41) which is 15 dBm. The single unit price for the low power version is $18 and the single unit price for the high power version is $25. They both contain an integrated antenna so pretty much all you have to do is supply 3.3 volts and data connections. The connections consist of asynchronous serial data using separate transmit and receive lines. Asynchronous serial data communication is built into most microcontrollers so integrating a Roving Networks module into a microcontroller project is pretty trivial.

The difficult part about using the Roving Networks modules is the actual electrical interfacing itself. Not only are the modules surface mount with fairly fine pitch connections, several of the connections are actually underneath the modules themselves. This means that soldering with conventional soldering iron techniques is out of the question. There are a number of techniques that will work with this part, including the toaster oven and hotplate methods, but if you want to avoid this altogether there are other (though somewhat more expensive) options available.

SparkFun Electronics[1] makes a number of breakout boards with Roving Networks modules on them. The current version of my Bluetooth prototype for TNC-X uses SparkFun Part Number WRL-10559, which contains the higher power RN-41 Roving Networks module. It's a little pricey (at $59.95) but it does work well. This break out board has four connections (RX, TX, power and ground). The only tricky part is that it is designed for both power and signals to be at 3.3 volts, not 5 volts (more on that below).

More recently, Roving Networks itself has begun to manufacture breakout boards at lower prices. The board containing the lower power Bluetooth module sells for $39, while the one with the higher power module goes for $45 (both quantity 1 pricing from Mouser[2]).

Interfacing the Roving Networks Modules

If you are interfacing these adapters to a 3.3 volt circuit, it's a pretty trivial job… power it up and throw serial data at it (the default baud rate is 115,200 baud) and you're in business. This is not RS-232 level serial data, of course, so you can't simply connect it to the serial port on a PC, for example. However, any standard microcontroller (such as the PIC in TNC-X) will do serial data at either 3.3 volts or 5 volts depending on the voltage you are supplying to the microcontroller.

TNC-X currently runs at 5 volts, however, and that requires some additional circuitry. First, the Roving Networks modules will not tolerate 5 volts so you must use a regulator to reduce the voltage to 3.3 volts. If you are using the Roving Networks break out board, it has a regulator on it that can reduce this to 3.3 volts so you won't need to add your own regulator. Since I was using the SparkFun module, I did need to provide regulation. I used a BA33BC0 3-pin unit. This is really overkill for this project because it will handle 1 amp of current and even the higher power RN-42 module draws under 100 ma, but I had a few of them lying around, so I used it.

Simply providing a regulated source of 3.3 volts does not solve the entire problem however, even if you use the Roving Networks module with the regulator built in. You still must deal with the problem that the microcontroller's transmitted data will be at 5 volts as well. Roving Networks recommends using a voltage divider circuit (see Figure 1). I used a 22K resistor in place of the 20K because I happened to have one lying around.

[1] www.sparkfun.com
[2] www.mouser.com

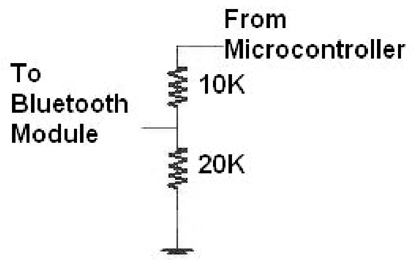


Figure 1

The voltage divider is placed on microcontroller's data transmit line to reduce the 5 volt signal to approximately 3.3 volts. The data line that transmits data from the Bluetooth module to the microcontroller can be connected directly since the 3.3 volts that comes out of the module will be properly recognized by the microcontroller.

Programming the Bluetooth Modules

The Roving Networks modules are highly configurable. To configure them you need to access their command line interface using a terminal. This can be, for example, a terminal program running on a PC, but if you do this, you must make provision for converting the RS232 level signals from the PC to 3.3 volt signals for the module. One possibility is to use a MAX232 level conversion chip to do this in conjunction with the voltage divider circuit above. Another alternative is to use a MAX3232 chip, which is a 3.3 volt version of the MAX232. This chip requires 5 .1 uf capacitors, but other than that you simply connect power and ground and the data lines and its ready to go. See Figure 2.

After you have connected the module to your PC, run a terminal program set up for 115,200 baud, 8 bits, no parity, 1 stop bit and no flow control. Power up the module and within 60 seconds type $$$. You'll get a command prompt and the device is ready to accept commands. All of the available commands are documented in the user manual.[3]

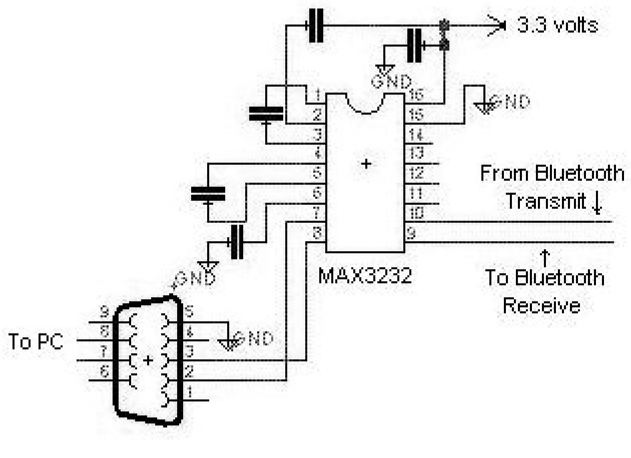


Figure 2.

[3] rovingnetworks.com/Docs/Bluetooth-RN-UM.pdf

A few of the commands you might find most useful are:

D	displays all of the basic parameter settings of the module.
SU	sets the baud rate. 115,200 is too fast for TNC-X so I reconfigured it to 9600.
SN	sets the name of the device. This is the name that comes up on your phone (or PC) to pair with. It's probably a good idea to change it to something other than the default, which is "FireFly", followed by a serial number.
SQ	This sets "special configuration commands".

Special configuration commands are useful because I found that for a TNC, it worked much better if it was configured for low latency data transfers rather than high throughput. The module is capable of much higher data rates than would ever be seen on a packet channel so sacrificing a bit of throughput was not a problem. To turn on this function, set the SQ value to 16.

Profiles

You may have noticed that Bluetooth is used for a lot of things these days. It can be used to control your mobile phone from your car (not just to transfer audio, but to initiate and answer phone calls as well). It can be used to access the address book on your mobile phone. It can be used to run wireless headphones or speakers. It can be used for keyboards and mice on computers. And much more. You may have also noticed that (unlike USB) when you buy a Bluetooth device, you don't have to load a device driver on your phone or PC. This is because Bluetooth has pre-written "profiles" which define the standards to which a particular type of Bluetooth device must conform.

The Roving Networks Bluetooth modules support only two of these protocols: the serial port protocol and the dialup networking protocol. The default is the serial port protocol, which is exactly what is needed for most microcontroller projects. With this protocol data that comes in one end shows up at the other side. When paired with a PC, it will show up as a standard COM port.

Interfacing with TNC-X

As noted above, to interface with the Roving Networks modules, all you need is power, ground, and receive and transmit data lines. TNC-X is a KISS mode TNC that was designed to be expandable.[4] It contains an expansion header which makes all of these signals available. So it was a relatively simple matter to design a plug in daughter board that provides Bluetooth capability to the TNC. The connections are as follows:

[4] For a complete description see Hansen, John A., "TNC-X: An Expandable Microcontroller-Based Terminal Node Controller". *ARRL and TAPR 23rd Digital Communications Conference Proceedings 2003* (Tucson, TAPR, Inc., 2003). Also available on www.tnc-x.com.

Header Pin	Function
1	Data from the TNC to the Bluetooth Module
3	Data from the Bluetooth Module to the TNC
7	Ground
8	+5 volts

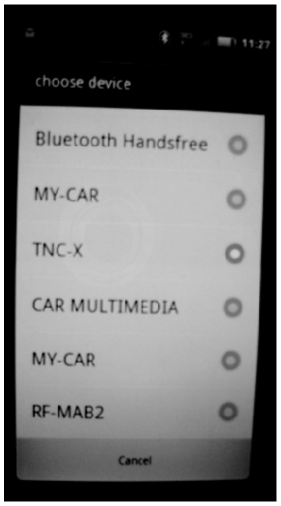
Because the power supply is 5 volts, it was necessary to provide the regulator and voltage divider circuits described above. It is anticipated that an upcoming version of TNC-X will move to 3.3 volts rendering this unnecessary. No changes need to be made to TNC-X itself (aside from removing the two jumpers to expose the 8 pin expansion header). The TNC was designed with exactly this sort of connection in mind.

Interfacing with a Phone

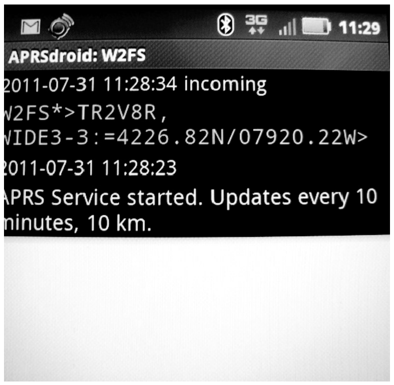
I am just beginning to get up to speed on mobile phone programming, so I decided to use off the shelf programs to test the Bluetooth connection rather than write my own. My phone is an Android powered Droid X. The first thing I needed to do was to pair my phone with the TNC-X. I did this the same way I did for any other Bluetooth device, by scanning for Bluetooth devices and clicking on TNC-X when it showed up on the list of found devices. The process went without a hitch and, somewhat surprisingly, didn't even ask me to input a pairing code.

The first program I tried with the Bluetooth connection after the phone was paired was a plain vanilla terminal program called "TerminalBT". This program was designed to allow the user to communicate over a Bluetooth serial port. Because TNC-X is expecting KISS formatted data, the incoming data over the Bluetooth connection was going to contain a lot of garbage characters, and it would not be possible to test data communication in the other direction. But it was able to determine whether or not data would flow between the TNC and the phone. I fired a couple of APRS packets at TNC-X and indeed they did show up on my phone's screen.

For APRS I used a program called APRSDroid written by Georg Lukas, DO1GL. It can be found at www.aprsdroid.org or in the Android Market. This program has been in development for a while and was used mostly by hams to insert their callsigns and locations in the Internet APRS datastream. Not much had been done on the RF side because it was not previously possible to connect these phones to existing TNC's. I contacted Georg and told him I was developing a Bluetooth daughter board for TNC-X and wanted to use his software with it. Since then he has rewritten some of the code to fully support KISS mode Bluetooth TNCs and it now works really well. APRSDroid is released under the GPL and written in Scala.

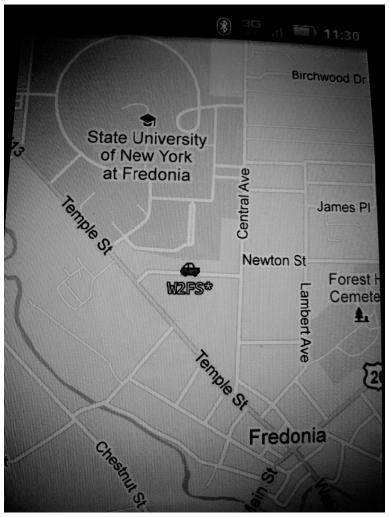


APRSDroid: Select TNC-X to connect via Bluetooth.



```
APRSdroid: W2FS
2011-07-31 11:28:34 incoming
W2FS*>TR2V8R,
WIDE3-3:=4226.82N/07920.22W>
2011-07-31 11:28:23
APRS Service started. Updates every 10
minutes, 10 km.
```

APRSDroid: Incoming position data from W2FS.



APRSDroid: Display W2FS position on map.

In addition to connecting the phone to the TNC so that data can be sent out over RF, it also will receive incoming data and either display the data or plot it with the Android's mapping software. The program interfaces with the phone's GPS receiver to obtain current location information.

Interfacing with a Laptop PC

A lot of laptops have Bluetooth these days so I wanted to try interfacing TNC-X with Bluetooth to my laptop. While I was able to get it to work, it's not clear how much of an advantage this is, since connecting the TNC to my laptop via USB will also power the TNC, while connecting via Bluetooth does not. However it is possible to connect to a laptop this way (mine is a Sony VAIO "Z" series running Windows Vista).

I did run into one issue, however. The software I used to manage laptop Bluetooth connections is provided by Sony. The first time I paired the Bluetooth module with the computer it all went quite well. I wanted to repeat the process so that I could better document it, so I tried deleting the TNC-X connection from the list of Bluetooth objects on the laptop and then redoing the connection. I don't think the software fully deleted the connection, though, because after pairing it, the Bluetooth module sent a request to connect to the laptop (this did not happen the first time). Somehow it didn't pair properly. I did a system restore to the point before I had paired it the first time and tried pairing again and this time it worked perfectly. I'm not certain whether this is a bug in the Sony configuration software or is an issue more widespread than just these Sony laptops. In any case, it was possible to get this connection up and going on Bluetooth as well.

Sheilding

TNC-X is fully enclosed in a steel box. When I designed the box, I didn't consider the possibility that I might want to connect to a computer or phone via Bluetooth, so I concluded the more shielding of TNC-X's internal oscillators the better. As a result, when I added the Bluetooth module I was concerned that the shielding might prevent the 2.4 GHz Bluetooth signal from connecting to devices outside the box. This caused me to select the higher powered Roving Networks module for this project rather than the low power one. As it turned out this was not a problem. TNC-X connects very well to my phone even with the box top in placed and secured by four screws.

Conclusion

Many years ago Mike Musick, N0QBF developed a program called “PocketAPRS” which was an APRS application that could work with Palm Pilots. This was back in the day when Pilots synced to computers using a computer serial port. Mike used this same serial port to send and receive data from a TNC. The days when mobile devices had serial ports are now long gone. The connectivity of choice for mobile devices is clearly now Bluetooth. Fortunately interfacing a TNC to a phone or a computer via Bluetooth is relatively simple. In addition, the cost is fairly modest and it is likely to become even cheaper going forward.