Implementation of a New Wireless Module for the Cornell Ranger

By
Andrew Mui
axm2@cornell.edu

Autonomous Walking Robots Team
School of Engineering
Cornell University

May 21, 2007

Faculty Advisor: Professor Andy Ruina,
Theoretical & Applied Mechanics

Credit Option: 3 Credits for Fall 2006
Also for fulfillment of technical writing requirement

TABLE OF CONTENTS

1 Introduction

Last December, the Cornell Ranger, which was designed by our team, the Autonomous Walking Robots Team, at Cornell University, walked just over one kilometer on an indoor track unassisted (with the exception of occasional steering controlled remotely to prevent it from running off the track). Since then, our team has been looking at ways to make the Ranger more power-efficient and robust (e.g. making it more resistant against falling due to changes in terrain or environment).

The motional behavior of the Ranger is controlled by a large set of around 60 parameters that control such values as how hard to push off with the ankles and how much power to put into each hip swing, among other values. The goal of our team is to determine the values of these parameters that give the Ranger the most power-efficient and reliable gait. Because of the large number of variables involved, finding the parameters for optimal performance will require hundreds of trial runs, each of which will require one or more parameters to be changed. In addition, in order to quantitatively look at how efficiently and reliably the Ranger is walking, data must be acquired from the Ranger so that it can be graphed and analyzed. Therefore, a fast and reliable two-way data transfer system is needed for two purposes: to send data to the Ranger for changing parameters (initial values) and to receive data from the Ranger for performance analysis.

For most of its walking life, the Cornell Ranger has been using a wireless module to acquire and transmit data. Specifically, the Radiotronix WI-232 DTS model has been used. The module currently attached to the Ranger daughterboard has been working since December 2006; however, the module had to be replaced four or five times before that. Therefore, we needed a more reliable wireless module so that we could spend more time analyzing data and less time replacing malfunctioning modules.

Data transfer rates have also been an issue. As more functionality is added to the Ranger (such as a feedback controller for fall protection), the number of control parameters, as well as the amount of data, will increase, and data will have to be sent in greater quantities. Overhead, caused in part by the packetization of data and error codes, among other things, is largely unavoidable, and further decreases the maximum rate at which data can be transferred without a high risk of data loss. With the current set of data, the Radiotronix module cannot reliably transfer data at a rate greater than 37.5 kBaud[1]. Further increasing the amount of data will cause an increase in the data transfer rate, and lead to more errors in the data. We would like to be able to send more data without increasing the likelihood that the data we receive will be corrupted.

This is where the MaxStream XBeePro module comes in. Though we do not know how reliable it is, we do know that it can transfer 60% more data than the WI-232 DTS per second[2], making it more useful when faced with greater data demands. Along with an increased maximum data transfer rate, the XBeePro has other advantages over its Radiotronix counterpart. These will be

[1] The Cornell Ranger processes, on average, 10 FFloat numbers every 16 milliseconds, or 625 FFloat numbers per second. Each FFloat number is 6 bytes, and each byte is 10 bits (when you include start and stop bits). This translates to a data transfer rate of 37.5 kBaud.

[2] The maximum data transfer rate for the XBeePro is 250 kbps [4]. It is 150 kbps for the WI-232 DTS [3].

discussed in this report. In addition, this report will describe the data transfer process between XBeePro modules, as well as the implementation of the XBeePro modules in the Cornell Ranger.

2 A Comparison of Wireless Modules

As already mentioned, the wireless module currently being used by the Cornell Ranger is the Radiotronix WI-232 DTS, and the module that will be used in the future is the MaxStream XBeePro. The following table compares several key specifications of both modules (values taken from [3], [4]).

	WI-232 DTS	XBeePro
Transmit Frequency	902-928 MHz	2.4 GHz
Line-of-Sight (LOS) Range	1500 ft.	1 mi.
Max. RF Data Rate	152.34 kbps	250 kbps
Receiver Sensitivity	-104 dBm*	-100 dBm
Transmit Current	57 mA**	215 mA**
Idle/Receive Current	20 mA**	55 mA**
Transmit Output Power	12 dBm*	18 dBm

Notes:
* – Measured at 2400 baud
** – V_{DD} = 3.3 V

Table 1 Comparison of WI-232 DTS and XBeePro Key Specifications

Although the XBeePro uses significantly more current (and therefore, more power) than the WI-232 DTS, it can transmit about 60% more data than its counterpart. In addition, the transmit output power of the XBeePro is 6 dBm higher, which translates to a factor of four, and the range of the XBeePro is significantly larger as well.

Another advantage that the XBeePro offers is that it is a plug-in module, while the WI-232 DTS is not. We do not know if the XBeePro is more reliable than the WI-232 DTS; however, we know that if the XBeePro unexpectedly fails or needs to be replaced, putting in new modules will be easier. Since the WI-232 DTS modules had to be soldered onto the daughterboard, the connections on the daughterboard had to be tested after installation of the module to ensure that no good connections were destroyed or that there were no short-circuits in the wiring. On the other hand, XBeePro replacement requires only two simple steps: removing the old module from the plug-in board and putting the new module into the plug-in board. No soldering is required; therefore, there is no chance of messing up any other part of the daughterboard in the replacement process.

3 XBeePro Data Communication

The following block diagram shows how data is transferred between the PC and the Ranger [5]. For convenience, the module connected to the PC will be referred to as the "PC XBeePro" and the module connected to the Ranger daughterboard will be referred to as the "board XBeePro".

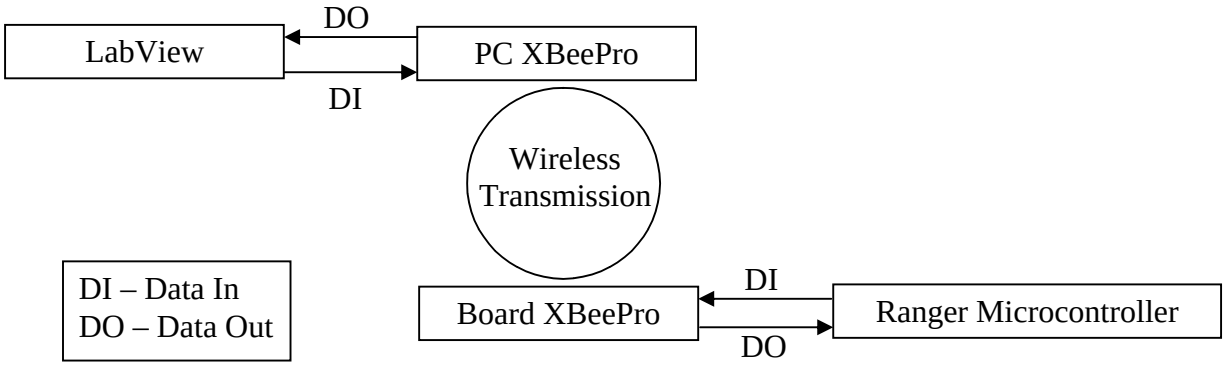

Figure 1 Dataflow Between Labview and the Cornell Ranger

As shown from the diagram above, data can either be sent from the PC to the Ranger (the forward link) or from the Ranger to the PC (the reverse link). In forward link communications, data from LabView (the application used to send data to and receive data from the Ranger) enters the data in pin of the PC XBeePro (see Section 6.1 for pin diagrams of the XBeePro), where it is transmitted through the air to the board XBeePro. The board XBeePro then sends the data through its data out pin to the Ranger's microcontroller. In reverse link communications, data from the Ranger microcontroller enters the data in pin of the board XBeePro, where it is transmitted through the air to the PC XBeePro. The PC XBeePro then relays the data through its data out pin to LabView, where the Ranger data can be collected and plotted for analysis.

But how does the XBeePro connect to the PC or the daughterboard in the first place? For the PC connection, the XBeePro plugs into a board with a USB connector, which subsequently plugs into the USB slot on the PC. For the daughterboard connection, the XBeePro plugs into an adaptor board, which can then be plugged onto the daughterboard. See Sections 6.2 and 6.3 for images and pin diagrams of the USB connector board and the adaptor board.

4 Implementation of the XBeePro Modules in the Ranger

4.1 Preparing the Modules for Communication

Before the modules could be connected to the Ranger daughterboard, we first had to make sure that the modules could "talk" to one another (i.e. send and receive data). In order for data transfer between the modules to occur, two conditions had to be met: 1) The baud rates of the two modules had to match, and 2) the destination address of the PC module had to be the same as the source address[3] of the board module, and vice versa [5].

[3] Each module has a destination address and a source address (both 16-bit binary numbers). The source address is the address of the source module (i.e. the address to which incoming data should be sent). The destination address is the address of the destination module (i.e. the address to which outgoing data should be sent).

The default baud setting on the modules was 9600 Baud [5]; however, we changed it to 115.2 kBaud so that the modules could handle the 37.5 kBaud data transfer rate currently being used by the Radiotronix modules plus overhead. The destination and source addresses for the board module are 0x1A0D and 0x0000, respectively. The source and destination addresses for the USB module were then changed accordingly so that the two modules could communicate.

The baud and address settings on the XBeePro modules can be changed via the program X-CTU. See Section 7.2 for information on using X-CTU to change module settings.

4.2 Echo Test

The echo test was used to determine if the XBeePro modules were communicating with each other by seeing if a message sent from the PC module via X-CTU was "echoed" back on the PC.

For the test, one of the modules was connected to the USB plug-in board, which was plugged into a PC. The second module, which was to be installed in the Ranger, was connected to the 232 Adaptor board. The VIN and GND connections were then made (pins 2 and 4, respectively, in the J1 section on the adaptor board). The VIN pin was connected to a 6V power supply. Finally, the DOUT and DIN pins (pins 1 and 3, respectively, in the J3 section on the adaptor board), are connected. By connecting the DOUT and DIN pins, any data sent from the PC module to the board module would be immediately routed back to the PC module. See Section 6.3 for pin diagrams of the adaptor board.

Once the baud and addresses of the two modules were set up properly (see Section 4.1), the echo test could be run. See Appendix B for information on how to send a message from an XBeePro via X-CTU. Below is a screen print of the results of the echo test.

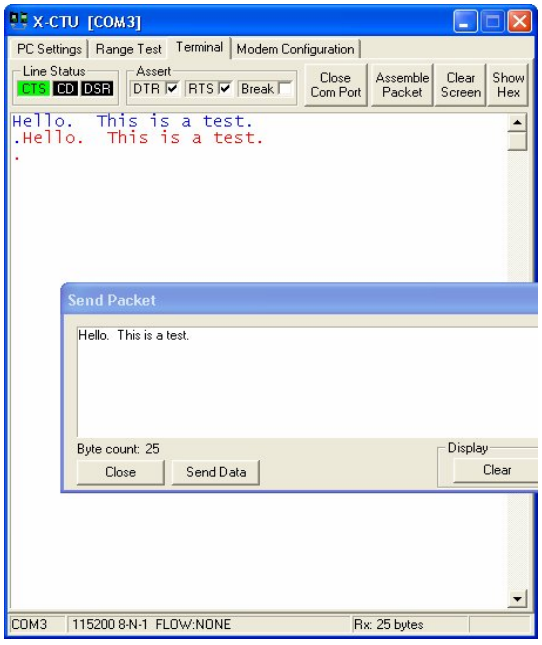

Figure 2 Echo Test Results

The blue text is the original message sent from the PC XBeePro, and the red text is the message received by the PC XBeePro after being routed through the board XBeePro. It can be seen that the echo test was successful, since the data sent by the PC was echoed back on the screen.

4.3 Preliminary Ranger Data Communications Test

After it was discovered that the Radiotronix modules were failing frequently, an RS232 serial port (see Section 8 for more information on RS232 serial ports) was put in so that the laptop could communicate with the Ranger even if the module did not work. For our first communications test with the XBeePro modules, the RS232 serial port was used so that we could externally connect the board XBeePro to the Ranger.

For the test, one of the modules was connected to the USB plug-in board, which was subsequently plugged into the laptop running LabView. The second module, which is to be installed in the Ranger, was connected to the 232 Adaptor board. The VIN and GND connections were then made (pins 2 and 4, respectively, in the J1 section on the adaptor board – see Section 6.3 for pin diagrams for the 232 adaptor board).

Next, the DIN and DOUT pins (pins 3 and 1, respectively, in the J3 section of the adaptor board) were connected to the Receive and Transmit pins (pins 2 and 3, respectively) on the RS232 port. An additional ground connection was also made between the adaptor board (pin 7 in the J1 section) board module and the ground connection on the Ranger (accessed via the ground pin (pin 5) on the RS232 port).

Once these wires were connected, the RS232 connector was plugged into the Ranger, and the Ranger code was changed[4] to use the module connected to the RS232 (so that the XBeePro, and not the WI-232 DTS, which was attached to the daughterboard, would receive and transmit data). We then set the LabView program to locate the USB XBeePro, and ran the data acquisition program.

For the most part, the data values received from the Ranger seemed to be accurate. However, we noticed one major problem: several of the parameters were giving infinite values, which we knew was wrong, since none of the parameters were supposed to be infinite in value. We then checked data log files generated by the LabView data acquisition program, and found that some data was missing. The program was written so that the logfiles would record a value once every 16 milliseconds. However, several of the entries were 32 or 48 milliseconds apart.

[4] In the C code for the Ranger, there are two parameters, WIRELESS and WIRECONN, which control the location to which data is sent on the Ranger. If WIRELESS is enabled, data will be sent directly to the wireless module inside the Ranger (the Radiotronix module). If WIRECONN is enabled, data will be sent through the RS232 port (which can be connected to an external wireless module or directly to the PC's RS232 port.

5 Summary and Conclusion

As of today, both XBeePro modules have been configured so that they can send data to and receive data from each other. If only one of the modules is sending data at a time (as was the case for the echo test), the data is transmitted without loss or corruption. However, if both modules try to send data at the same time (as was the case with the preliminary Ranger data communications test with the board module connected to the RS232 serial line), data loss and corruption occurs.

One possible solution to the data problem is to set up the handshaking connections, in addition to all the other connections already made for the preliminary Ranger test. Handshaking is a process where both devices involved let each other know when they are ready to send and receive data. If the handshaking connections are set up properly, there should never be a period where both devices are trying to send data at the same time.

On the XBeePro modules, handshaking is controlled by the active-low CTS (clear-to-send) and RTS (request-to-send) lines. RTS is an input line to the XBeePro, and comes from the Ranger microcontroller. CTS is an output line from the XBeePro, and goes to the Ranger microcontroller[5]. When RTS is low, the microcontroller is telling the board XBeePro that it (the microcontroller) is ready to receive data, and when CTS is low, the board XBeePro is telling the microcontroller that it (the board XBeePro) is ready to receive data.

The Radiotronix module uses the CTS line; however, since the RS232 port was originally used only for a wire-to-wire connection between the PC and the Ranger (in the event of a Radiotronix module failure), the handshaking lines were never connected on the Ranger's RS232 port. Therefore, in order to test the XBeePro modules externally with handshaking enabled, wires must be connected onto unused pins on the Ranger daughterboard for the CTS and RTS lines, and connected to the CTS and RTS pins on the RS232 port as well. In addition, the C code must be changed to utilize signals coming from these lines.

Hopefully, when the CTS and RTS lines are properly set up, the data problems will disappear, and the XBeePro modules will be able to send data faster and more reliably than the current Radiotronix modules.

[5] On the PC side, data goes between the PC XBeePro and LabView. On the Ranger side, data goes between the board XBeePro and the Ranger microcontroller. We don't need to worry about wire connections on the PC side, since the PC XBeePro is connected to the PC via the USB connector board.

6 Appendix A: Pin Diagrams and Pictures of Hardware Used

6.1 XBeePro Wireless Module Images and Pin Description

Below is an image of the XBeePro wireless module.

Figure 3 XBeePro Wireless Module

The following table is a pin assignment table for the XBeePro. This is an abridged list, and describes only the pins mentioned in other sections of the report. A complete pin description table can be found at [5]. Pin 1 is in the upper left hand corner of the module, pin 10 is in the lower left hand corner, pin 11 is in the lower right hand corner, and pin 20 is in the upper right hand corner of the module.

Pin Number	Name	Direction	Description
1	VCC	N/A	Power supply
2	DOUT	Output	Data Out
3	DIN	Input	Data In
10	GND	N/A	Ground
12	CTS_L	Output	Active-low clear-to-send flow control
16	RTS_L	Input	Active-low request-to-send flow control

Table 2 Pin Assignments for the XBeePro

6.2 XBeePro USB Plug-in Board Images

Below is an image of the USB Plug-in Board with the XBeePro wireless module plugged in.

Figure 4 USB Plug-in Board with Attached Module

6.3 XBeePro 232 Adaptor Board Images and Pin Diagrams

Below is an image of the 232 Adaptor Board (plug-in) for the XBeePro, as well as an image of the different segments of the board [1]. In Figure 6, the red boxes correspond to pin 1 of the corresponding segment. For the J1 and J5 segments, the left column of pins is odd-numbered, and the right column of pins is even-numbered. Note that the J3 section is comprised of the three connectors at the upper left hand corner of the board. Pin 1 in J3 is the leftmost pin, and pin 3 is the rightmost pin.

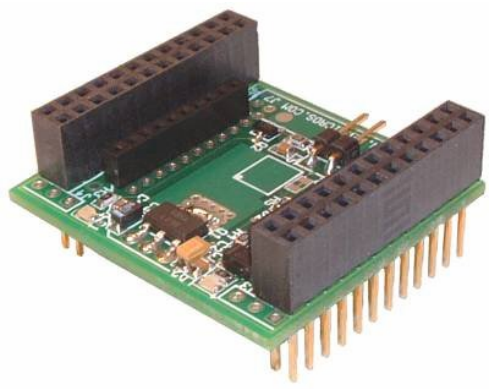

Figure 5 232 Adaptor Board

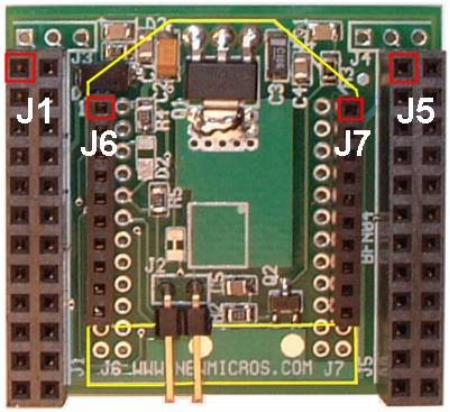

Figure 6 232 Adaptor Board Pin Layout

The table below shows several important pin connections for the 232 Adaptor Board. A complete list of pin connections can be found at [1]. The names of these pins correspond to the pins of the same names on the XBeePro.

Section Number	Pin Number	Name	Description
J1	2	VIN	Power supply, 6-12 VDC
	4	GND	Ground
	7	GND	Ground
J3	1	DOUT	Data Out
	2	GND	Ground
	3	DIN	Data In
J5	17	CTS_L	Active-low clear-to-send flow control
	18	RTS_L	Active-low request-to-send flow control

Figure 7 Pin Assignments for the 232 Adaptor Board

7 Appendix B: X-CTU Reference Guide

This section talks about how to use the X-CTU application to connect to an XBeePro wireless module, as well as how to configure settings on the module.

7.1 Establishing a Connection With an XBeePro Module

Before messages can be sent or settings on an XBeePro module can be changed or viewed, X-CTU must establish a connection between the serial port on the PC and the module. The first step in establishing this connection would be to plug the module into an XBeePro USB plug-in board, and plug the board into a USB slot on a PC running X-CTU. If the board is being plugged into the PC for the first time, a Found New Hardware message will appear, asking you to install some hardware to connect the USB and serial ports in the PC. Click on the "Install the Software Automatically" option.

Next, open X-CTU. The following window should appear.

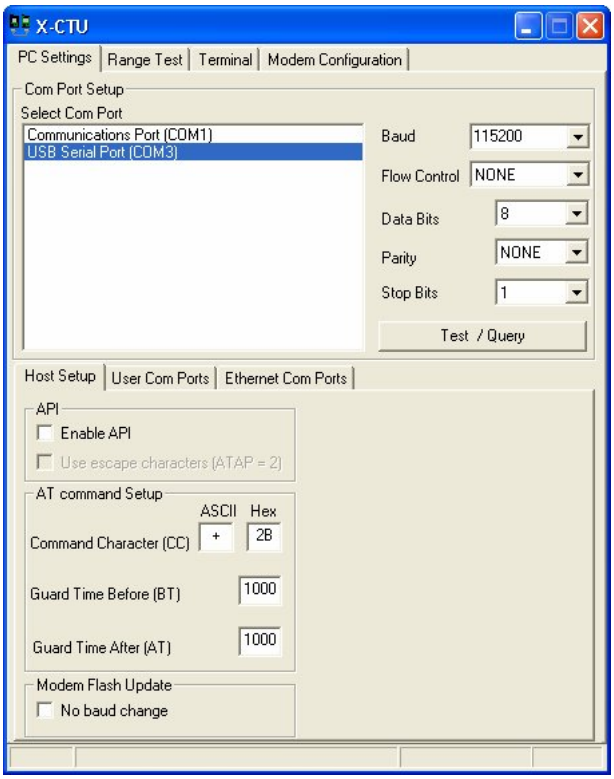

Figure 8 X-CTU, PC Settings Tab

You should be in the PC Settings tab. To connect to the XBeePro, click on "USB Serial Port" under the Select Com Port window, and click on the Test/Query button. At this point, you should either get a pop-up window that tells you that the communication with the modem is OK, or one that tells you that it was unable to connect to the modem. If you get the first message, the connection has been made. If, however, you get the second message, the Baud setting probably needs to be changed. The default value for the Baud setting is 9600 Baud; however, the XBeePro used here required a Baud setting of 115200 Baud.

7.2 Configuring/Viewing Parameters on the XBeePro Module [5]

Once a connection between an XBeePro module and X-CTU has been established (see Section 7.1), parameters on the module can be configured. First, click on the Terminal tab. The window should look like this:

Figure 9 X-CTU, Terminal Tab

At this point, the XBeePro module should be in idle mode, which means that it is not receiving or sending any data. Before any parameters can be sent, the module must be told to go into command mode. This can be done by typing in the command "+++" and waiting for an "OK" response. Once you get the OK response for the +++ command, if you wait for more than 10 seconds without typing in any more commands, the module will go back into idle mode, and you will have to re-enter the +++ command to set the module back into command mode. This 10 second period is known as the command mode timeout period, and can be set via the command ATCT (command mode timeout).

To send a command to the XBeePro in X-CTU, use the following format:

"AT" + (AT Command) + [Space] + [Parameter] + <CR>[6].

The letters "AT" tell the XBeePro that a command is being sent, and must be included as the first two characters in the command. The next two characters represent the AT command you want to send. All XBeePro commands have a unique two-character code. If you want to read the value currently being associated with that command, hit enter after entering the AT command. If you

[6] Square-bracketed commands are optional. <CR> - Carriage return.

want to write a new value to a command, hit the space key once, then enter in a parameter. The parameter entered must be a hexadecimal number (signified by the characters 0x preceding the number). The following figure shows several parameters on the XBeePro being configured and/or viewed.

Figure 10 X-CTU, Configuring and Viewing Parameters

The first AT command sent is BD, which sets/returns the Baud rate of the module. A value of 7 for this command means that the Baud rate is 115.2 kBaud. The next command is MY, which sets the/returns the source address of the module. As can be seen from the figure above, the source address was first outputted on the screen, then changed twice. The last command is WR, which allows you to write any changes made since the module was last plugged into the PC to the non-volatile memory of the XBeePro. Normally, whenever you make a change to a parameter, it gets stored in the volatile memory of the XBeePro, which gets cleared when the module is disconnected from the PC. Using the WR command allows you to set new default values for different commands. See [5] for detailed information about the names and formats of all commands supported by the XBeePro.

7.3 Sending Messages

Once you have established a connection between an XBeePro module and X-CTU (see Section 7.1), as well as a connection between the aforementioned module and another XBeePro module, you can send messages between the two. To send messages, first click on the Terminal tab, then click on the Assemble Packet button. The following window should pop up.

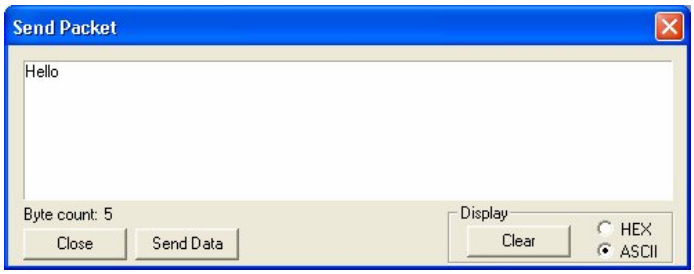

Figure 11 Sending Packets of Data in X-CTU

Once you have typed in your message, click on Send Data. The message should appear in blue letters in the Terminal screen.

8 Appendix C: RS-232 Port

The RS-232 is a standard for serial data connections between two devices. The 9-pin version of the RS-232 port is found on most PCs, and is also used on the Cornell Ranger as a way to connect wireless modules externally for testing, as has been done for the XBeePro modules. Alternatively, the port can be used for connecting a wire directly between the PC and the Ranger, so that data can be sent even if the module inside the Ranger fails. The following table lists the functions of each of the pins on the RS-232 port [2].

Pin Number	Pin Function
1	DCD (data carrier detect)
2	Receive Data
3	Transmit Data
4	DTR (data terminal ready)
5	GND
6	DSR (data set ready)
7	RTS (request-to-send)
8	CTS (clear-to-send)
9	RI (telephone line ring indicator)

Table 3 RS-232 Serial Port Pin Description

9 References

[1] Newmicros.com. “XBee-Adapter-232.” 17 May 2007 <http://www.newmicros.com>. Path: ZigBee; XBee-Adapter-232.

[2] “RS-232”. *Wikipedia: The Free Encyclopedia*. 12 April 2007. 17 May 2007 <http://en.wikipedia.org/wiki/RS-232>.

[3] “Wi.232DTS-FCC-R Datasheet.” *Radiotronix*. 17 May 2007 <http://www.radiotronix.com/datasheets/WI232DTS-FCCdatashortrev2.pdf>.

[4] “XBee OEM RF Modules Datasheet.” *MaxStream*. 17 May 2007 <http://www.maxstream.net/products/xbee/datasheet_XBee_OEM_RF-Modules.pdf>.

[5] “XBee/XBeePro OEM RF Modules Product Manual.” *MaxStream*. 17 May 2007 <http://www.maxstream.net/products/xbee/manual_xb_oem-rf-modules_802.15.4.pdf>.