**Invisible Robotics Bumper (iRoboBumper)**

**Software and Configuration**

Overview: This section will describe how to load the software created to run on the Arduino Pro Mini, and discuss how to configure the XBee wireless modules. As a note, version 1.0.6 of the Arduino IDE was used during development of the project as it was the latest version at the time, but likely any version will work. This Arduino IDE can be downloaded from Arduino's website at <http://arduino.cc/en/main/software>.

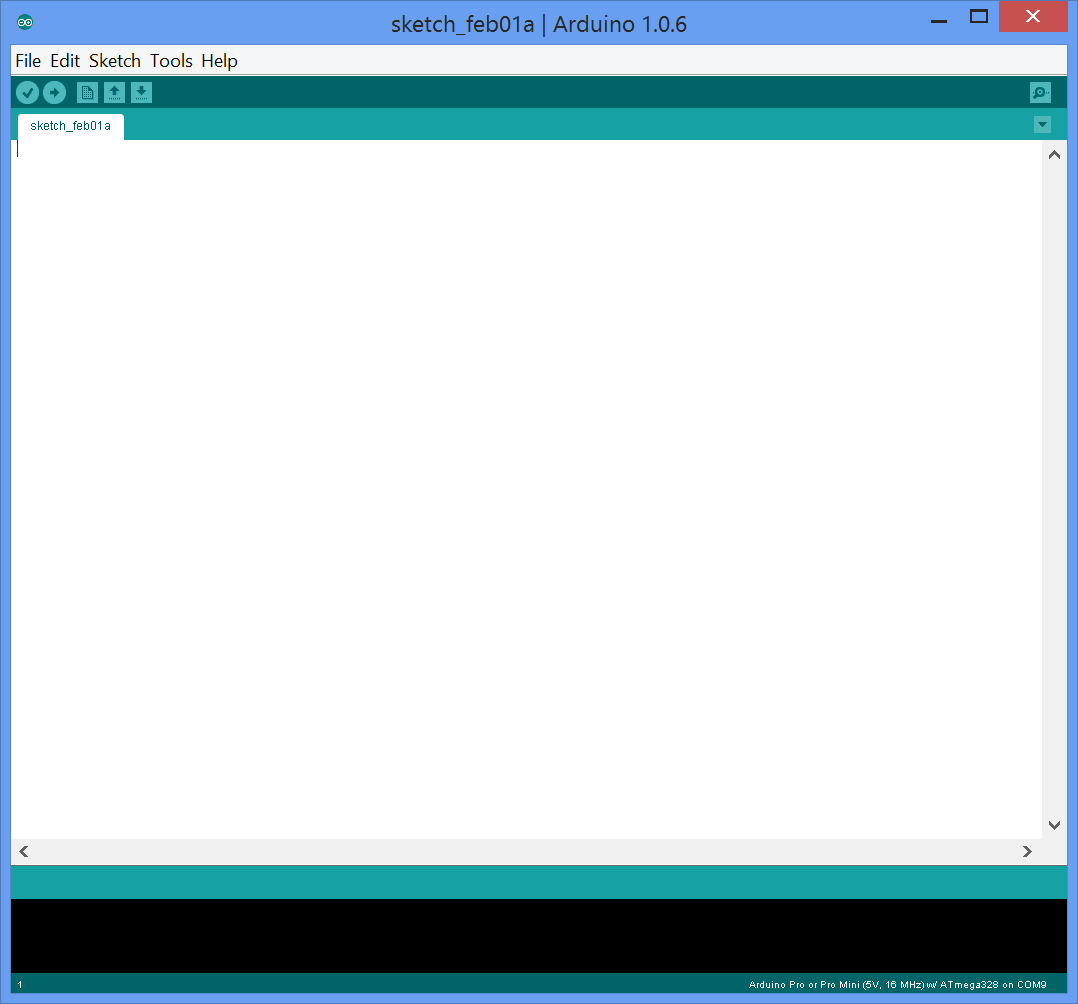
The Arduino sketch that was created for this project is contained in a single file, iRoboBumper.ino. <link to Source Files section in a new Window.> The software is well-commented to facilitate understanding of how it works and why the software was architected the way it was.

**Arduino Performance Issue Work-around:**

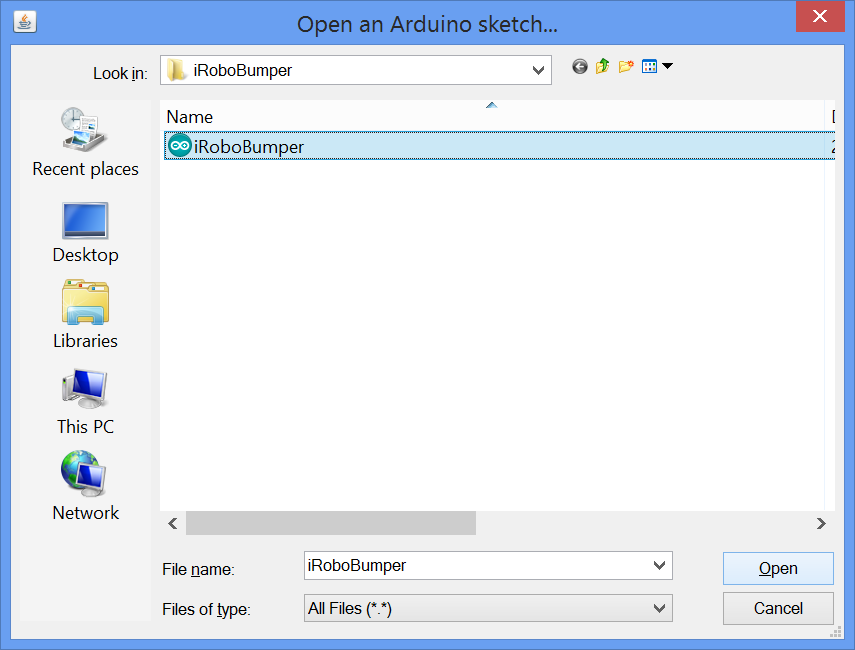
Some of the software uses Arduino functions, and so forth; however, other areas configure microcontroller registers directly in order to work around performance issues with Arduino. One example is with reading analog voltages. In this application, the proximity sensor needs to be read at a fairly high rate to ensure accurate timing measurement of the alignment holes. In addition, the temperature sensor and battery voltage need to be read periodically.

However, the Arduino function, AnalogRead, waits until the requested voltage has been read which is typically in the range of 100 microseconds. Unfortunately, when you want to make readings at a decent pace, these 100 microsecond blocks add up quickly. Instead, we configured the registers of the ATmega microcontroller to allow an interrupt to be generated when the analog to digital conversion has finished, and the Interrupt Service Routine (ISR) processes the data and starts the next conversion. This is just one example, and all of this is well-documented within the code.

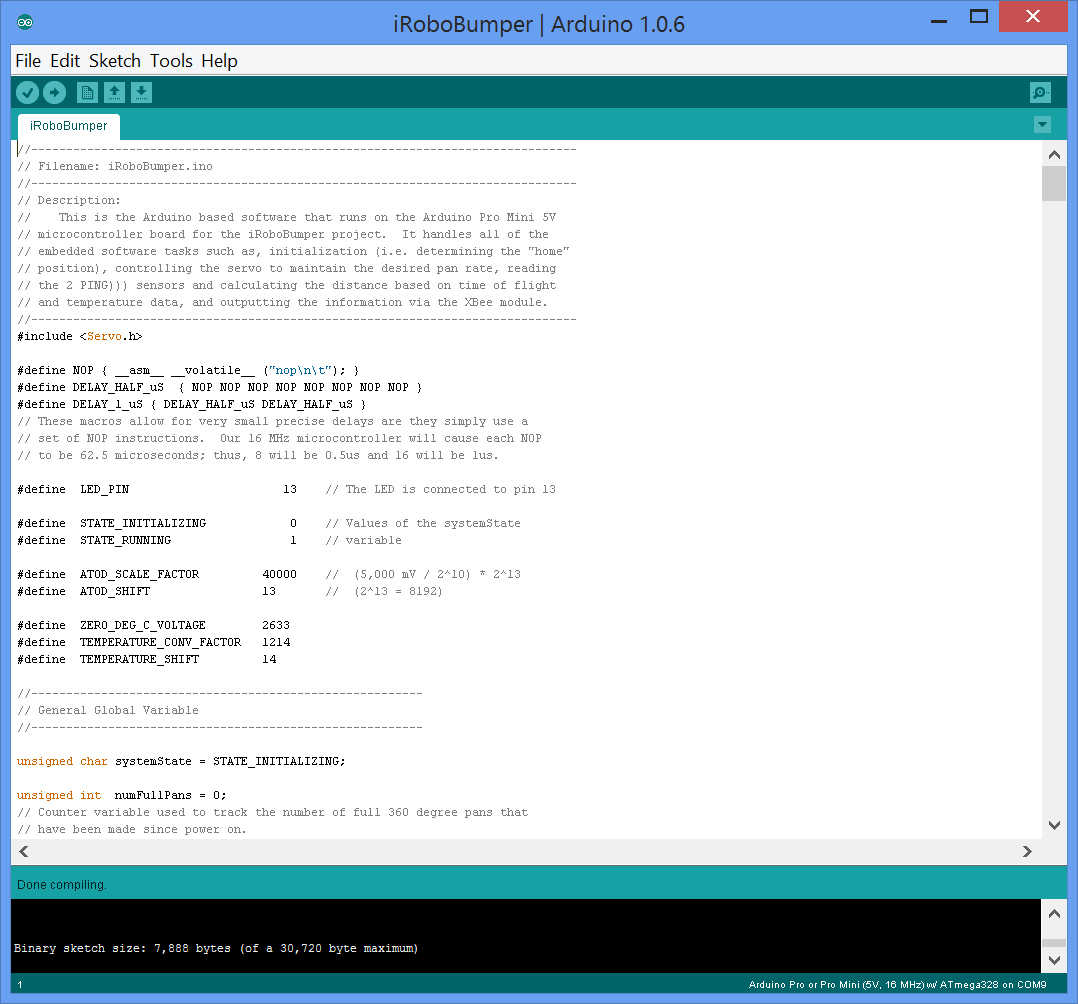
1. **Starting the Arduino IDE:**
2. Once you have the IDE installed, you will need to start the Arduino IDE. Once the application starts up you will be presented with the window shown in Figure 42.

  
  
Figure 42 - Arduino IDE Interface at Start Up

1. Select File->Open and navigate to the iRoboBumper folder where the *iRoboBumper.ino* file resides. Select the file and click open. Figure 43 shows the Open File dialog window prior to clicking open.

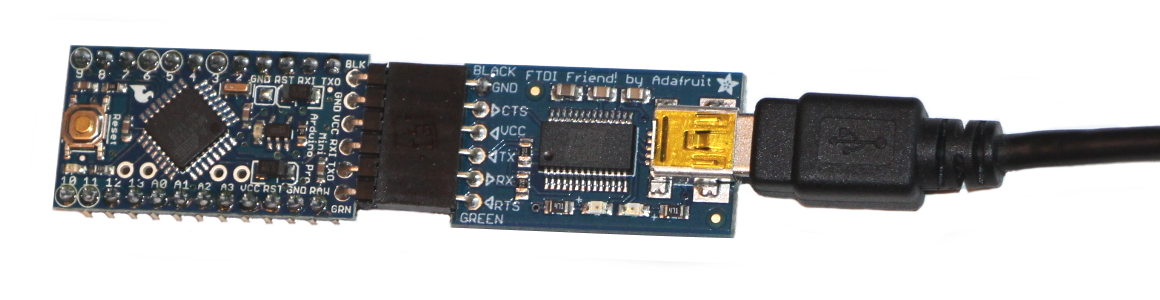
  
  
Figure 43 - Selecting the iRoboBumper.ino File

1. Once the Arduino IDE opens the iRoboBumper [sketch](http://arduino.cc/en/Tutorial/Sketch), you will see a window with the contents of the iRoboBumper file. Select *Sketch->Verify / Compile* from the menu bar and the sketch should build, as shown in Figure 44. Note: It shows this sketch is using 7,888 out of 30,720 bytes available. Make sure you select the PRO Mini (5V/16MHz) in the IDE before compiling.

  
  
Figure 44 - Successful Build of iRoboBumper.ino – click on image to download

<Click image to download this file: <fn≥ = fig44\_CompileArduinoSketch.png Located here: S:\Marketing Communications\TECHNICAL MARKETING\Content Team\Mouser OSHW Projects\Landa\Robotics - Project 1\images.zip

1. Next, connect your [Arduino programmer](http://www.mouser.com/access/?pn=992-ARD-PROGRAMMER) to the Arduino Pro Mini, and use a standard USB cable to connect the Arduino programmer to your PC. The Arduino Programmer (ARD-PROGRAMMER) is used to program the Arduino LilyPad, PRO and Pro Mini. This setup is shown in Figure 45, with the programmer sandwiched between the Arduino Pro Mini and a USB cable.

  
Figure 45 - Arduino Pro Mini (left) and the Arduino Programmer (right) Connected. A standard USB cable (far right) connects the Programmer to a PC.

1. The final step is to program your Arduino Pro Mini. Simply select *File->Upload* to have the [Arduino IDE](http://arduino.cc/en/main/software) and embedded bootloader transfer your program to the Pro Mini. The first time you use your programmer you will have to tell the Arduino IDE which COM port it is connected to, but you should be able to identify it from the list of available COM ports on your computer.
2. Once complete you should see a "Done Uploading" message in the blue status bar near the bottom of the IDE window. Now you can disconnect the USB cable from the programmer and then separate the programmer and the Arduino Pro Mini. Finally, plug your Arduino Pro Mini into the socket on the Main System PCB and we are almost ready to power up!
3. **Configuring the XBee Wireless Modules**

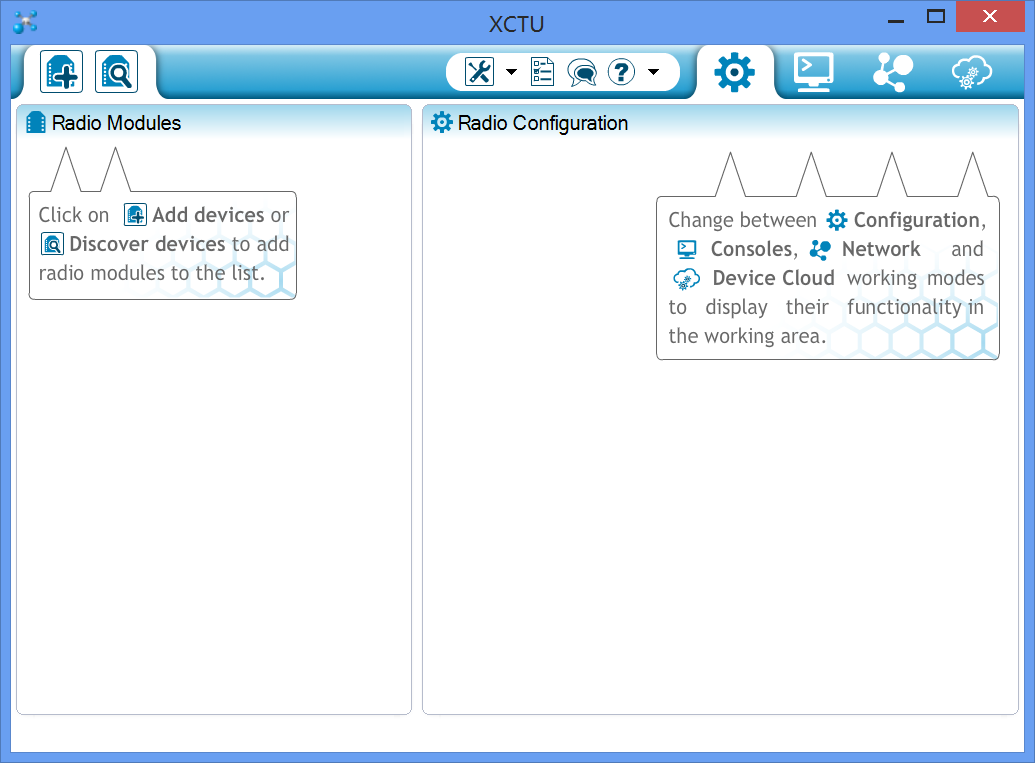
Before we power up the system, we need to configure the XBee wireless modules to act as wireless serial port bridges. During development both WiFi and ZigBee-based XBee modules were successfully tested. However, the intention is that this will be short range wireless communication to a module located on the host robotics platform (i.e. likely less than a foot or two away). In this case, the lower power ZigBee-based modules make the most sense and will be used to demonstrate the system.

1. To configure the XBee modules you need to connect to their serial interface using some type of adapter. In the Bill of Materials (BOM) (located in the Table of Contents for this project online), there are USB-to-XBee modules that are used for this exact purpose. Figure 46 shows the two ZigBee-based wireless modules plugged into the USB to XBee adapters and USB cables connecting them to the PC.

  
Figure 46 - XBee Modules on USB to XBee Adapters

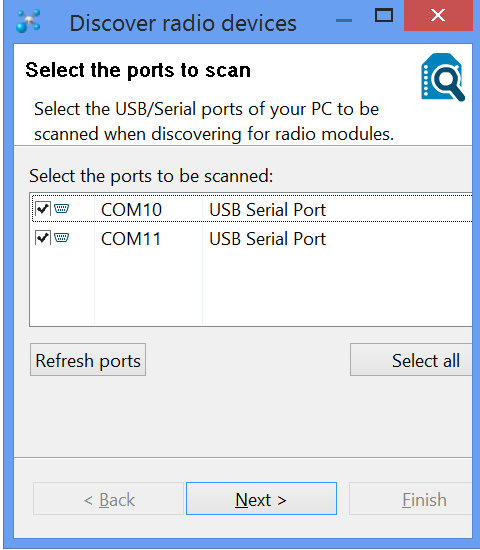
b) **XBee Configuration Tool:**

Digi International, the manufacturer of the XBee modules has an XBee configuration tool available called XCTU. You can download it from Digi's website at [http:\\digi.com\xctu](http://digi.com/xctu). Once installed, we will use this application to configure the two XBee modules to act as a wireless serial adapter so that our system and the host platform can communicate while our system continuously rotates. To start, run the XCTU application. Once it has started, you will be presented with the screen shown in Figure 47.

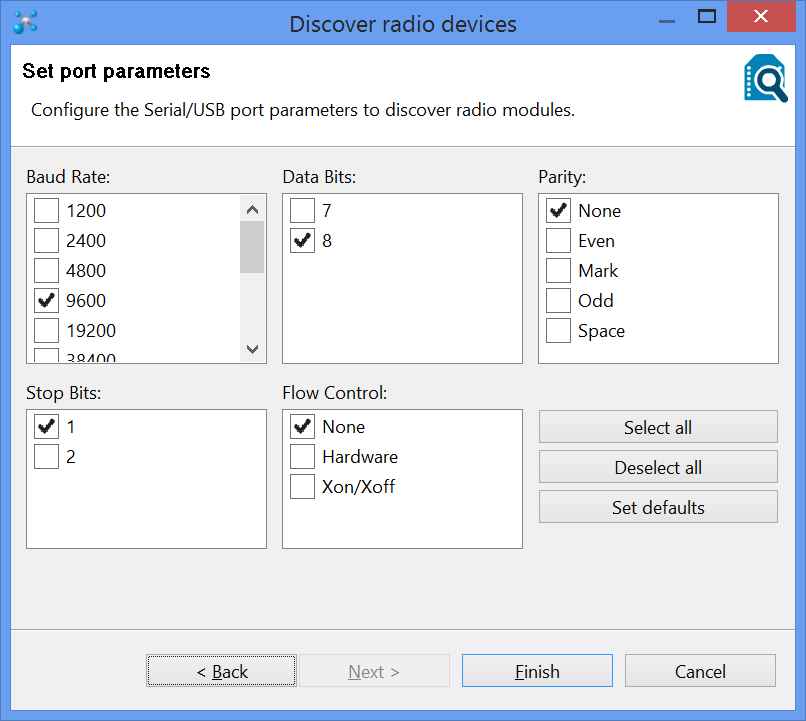
  
Figure 47 - XCTU Startup

1. Next, you will want to click on the "Discover devices" button. It is the icon on the left side of the menu bar that looks like an XBee module with a magnifying glass over it (Figure 47.)

c) Once you click it, XCTU will present you with the window below and ask what COM port you want it to discover XBee modules on. In Figure 48, both of the ZigBee modules were plugged in and they enumerated as COM10 and COM11. Thus, we have selected to have XCTU discover modules on each of these ports.

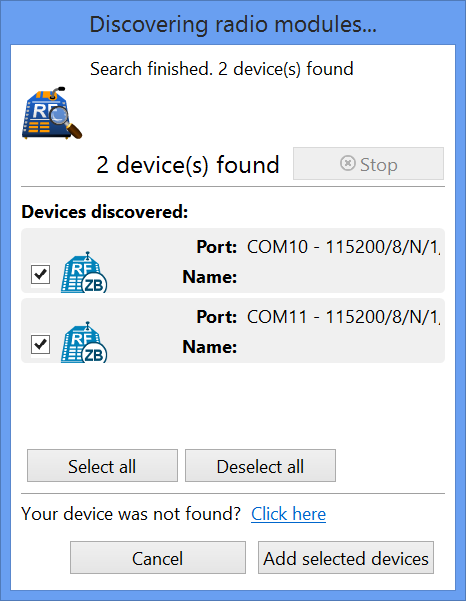
  
Figure 48 - XCTU COM Port Discovery Selection

d) Next, XCTU will allow you to select what serial port settings it should use to discover the XBee modules. Figure 49 shows the default selections which should allow XCTU to discover your modules. Note: if you change the baud rate setting of the module, as we will do shortly, and want to use XCTU in the future, you will need to modify these settings.

  
Figure 49 - XCTU COM Port Settings for Discovery

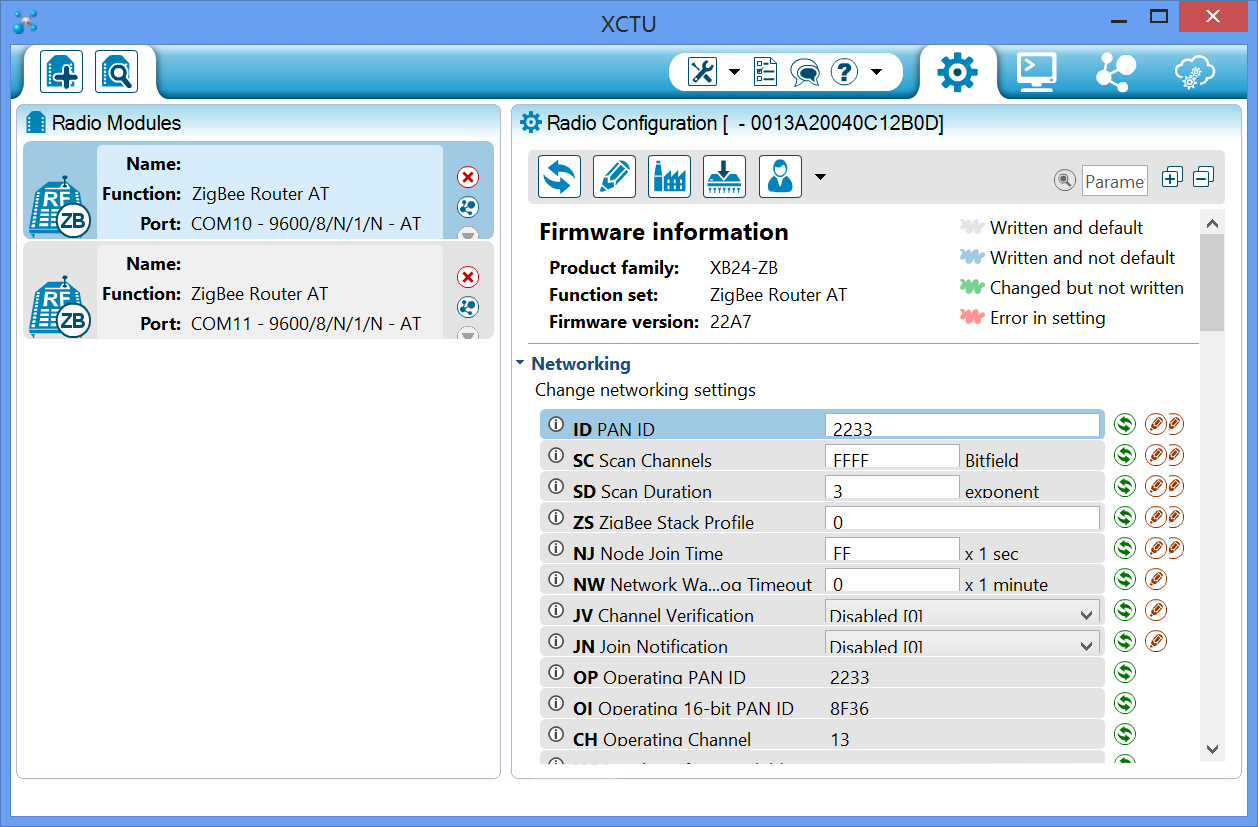
Once you have selected the settings for XCTU to use, click 'Finish'.

e) Next, XCTU will attempt to communicate with the modules on the COM ports you have selected. This will take some time depending on the number of ports and settings you selected. Once complete, XCTU will present you with the dialog window shown in Figure 50 and ask you which of the discovered modules you would like to add to the 'Radio Modules' list. Select both of your modules and click on the 'Add selected devices' button.

  
Figure 50 - XCTU Discovered Modules

Now, you will be back to the main XCTU window and your wireless modules will be listed on the left.

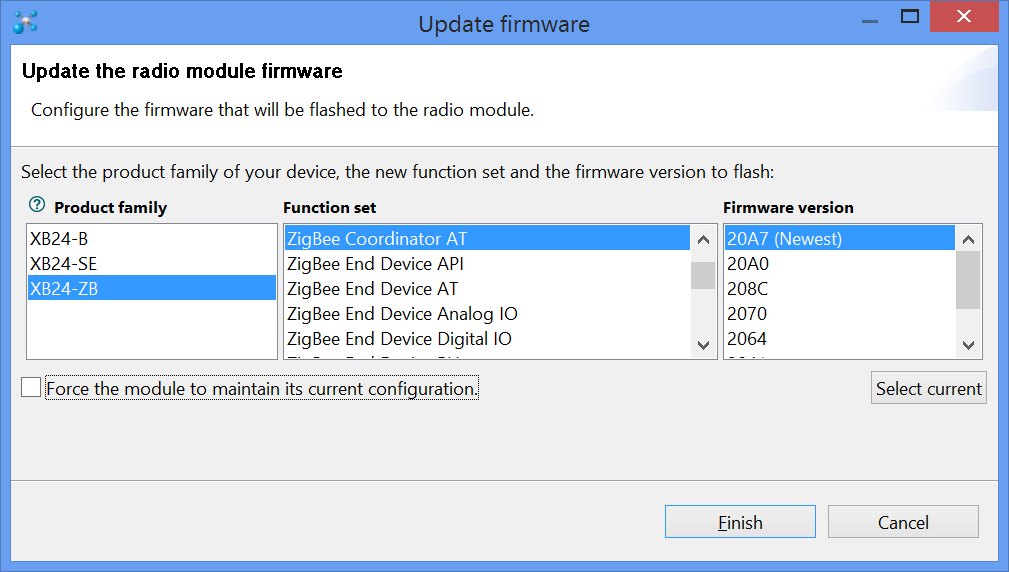
1. In Figure 51, back to the main XCTU window, you can see the two modules which are connected to COM10 and COM11.

  
Figure 51 - XCTU Application with Discovered Modules

You will also notice that both of the discovered modules are currently configured to be ZigBee Router AT devices, as this is how they were configured from the factory. So the first thing we must do is to change the firmware in one of the devices to be a *ZigBee Coordinator AT device*.

g) To update the firmware in a module, click on the module under the 'Radio Modules' list on the left side of the application.

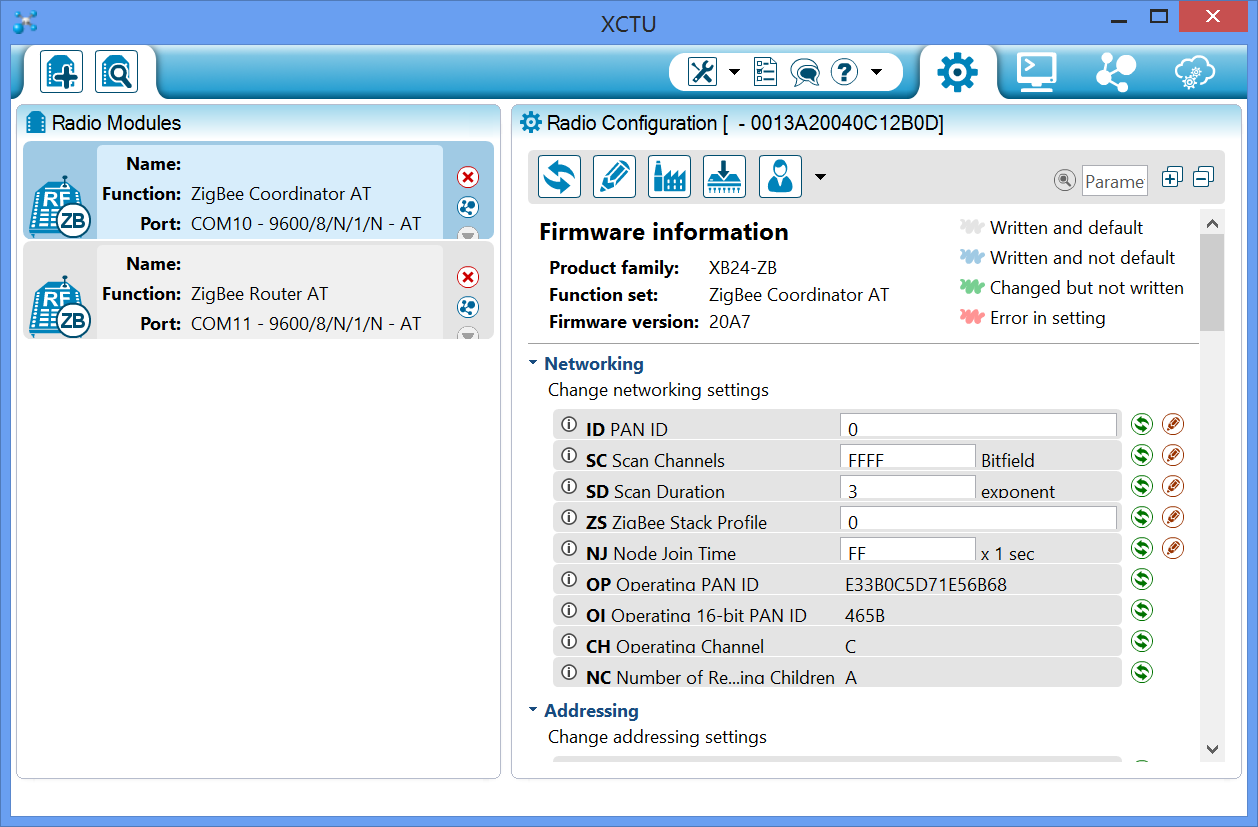
h) Next, click on the 'Update firmware' button which is on the right side of the application and looks like an electronic component with an arrow pointing down. Clicking on it will cause the 'Update firmware' dialog box to appear as seen in Figure 52.

  
Figure 52 - XCTU Update Firmware Dialog

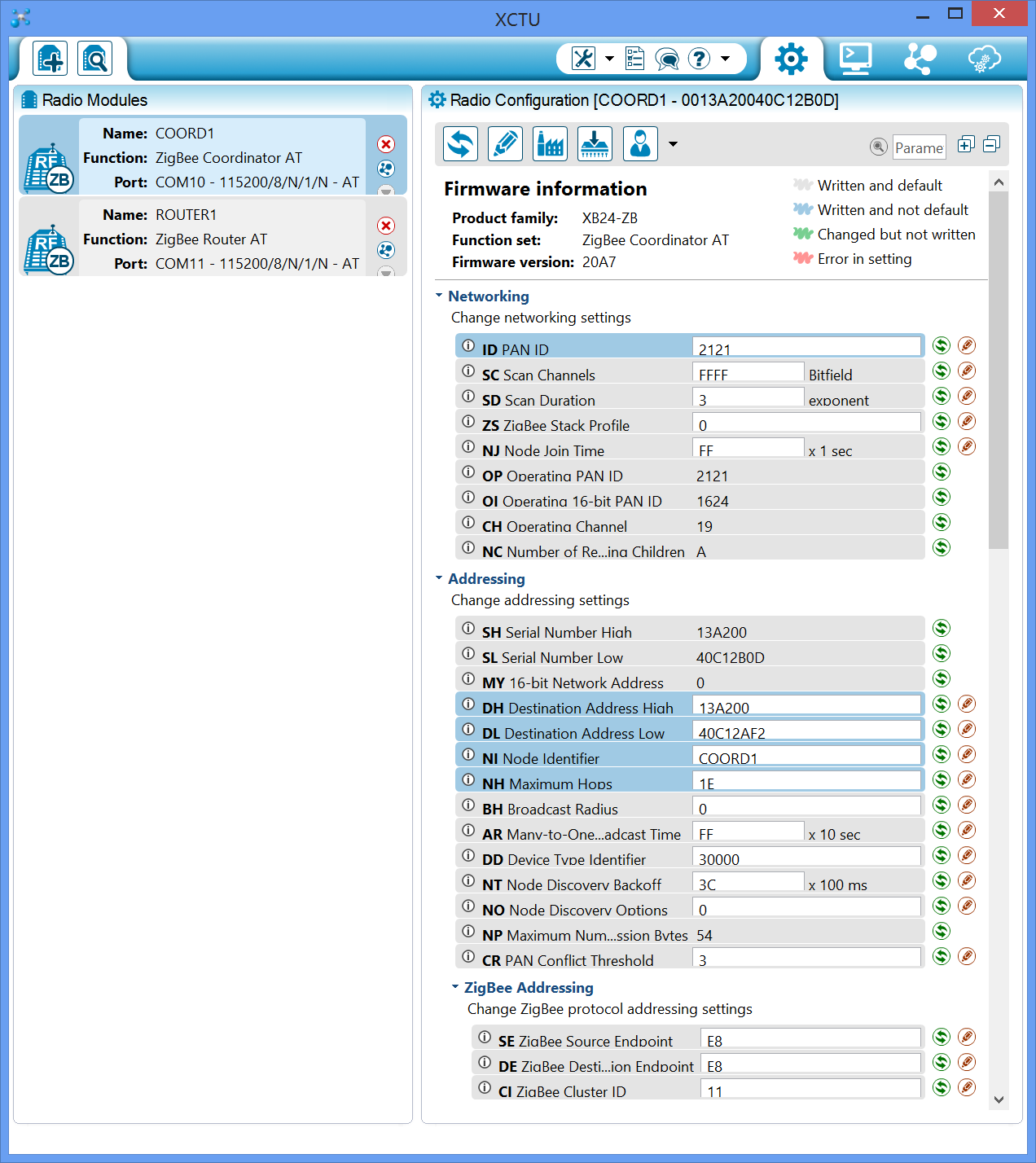
1. Next, select the XB24-ZB option under *Product family*, then the select 'ZigBee Coordinator AT' under *Function set*, and finally select 20A7 (Newest) under Firmware version.

These are also the options that were selected in Figure 52.

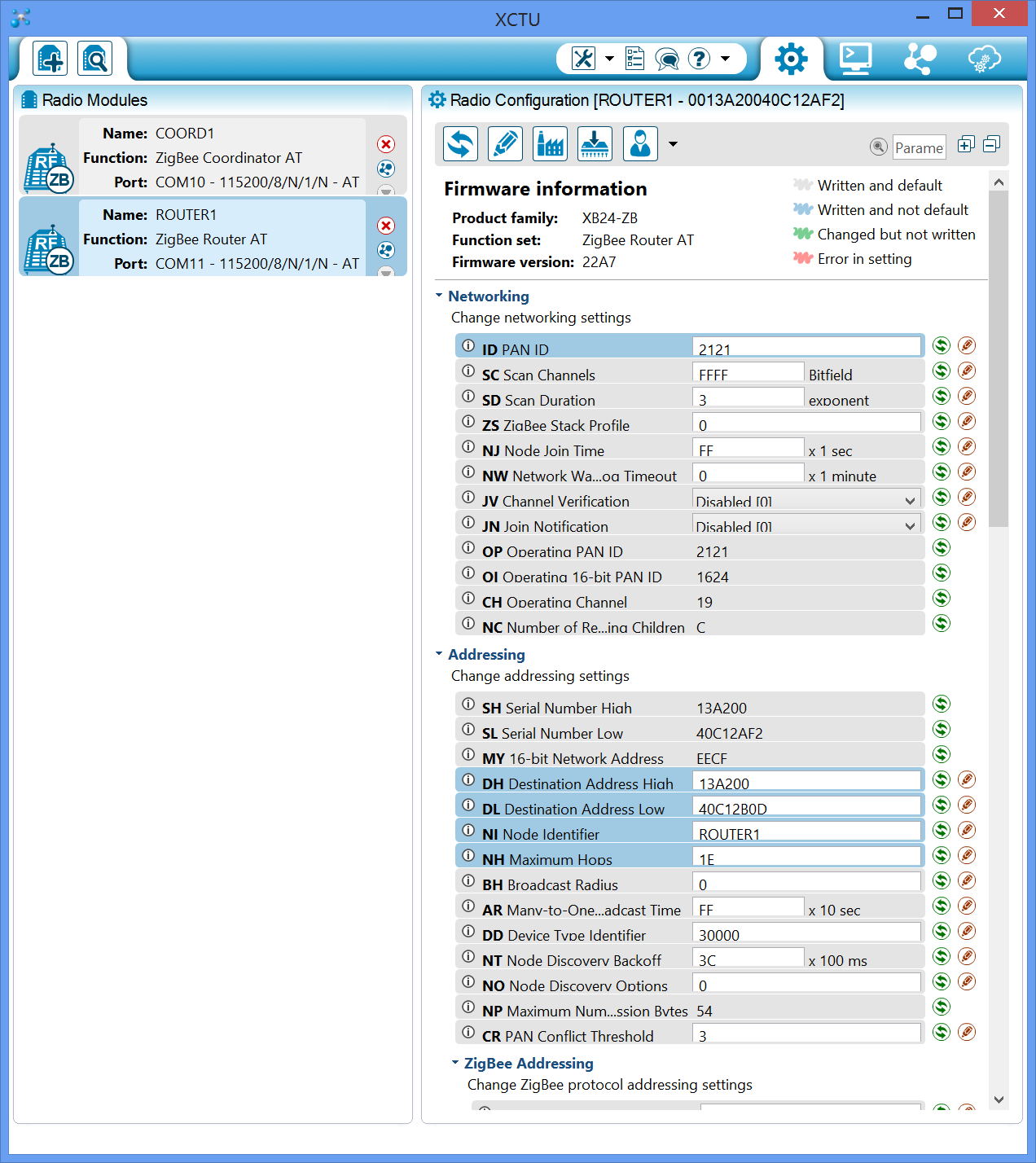
1. Now click 'Finish' to have XCTU update the firmware on the selected module. After the module has been updated, XCTU will return to the main window and the module is now configured as a ZigBee Coordinator AT as can be seen in Figure 53.

  
Figure 53 - XCTU Application after Firmware Update

1. We need to update the settings of both devices to allow them to act as a point-to-point wireless serial port adapter. The Coordinator AT needs to have its PAN ID set to a unique value; 2121 was the value chosen here, but yours can be different. Just remember the value as the Router's PAN ID must be the same!
2. The Destination High and Low Address (DH and DL) must be set to the Serial Number High and Low values (SH and SL) of the Router. In addition, you can set the Node Identifier to a value, but this is not required. All of these values can be seen in Figure 54.

  
Figure 54 - ZigBee Coordinator AT Settings

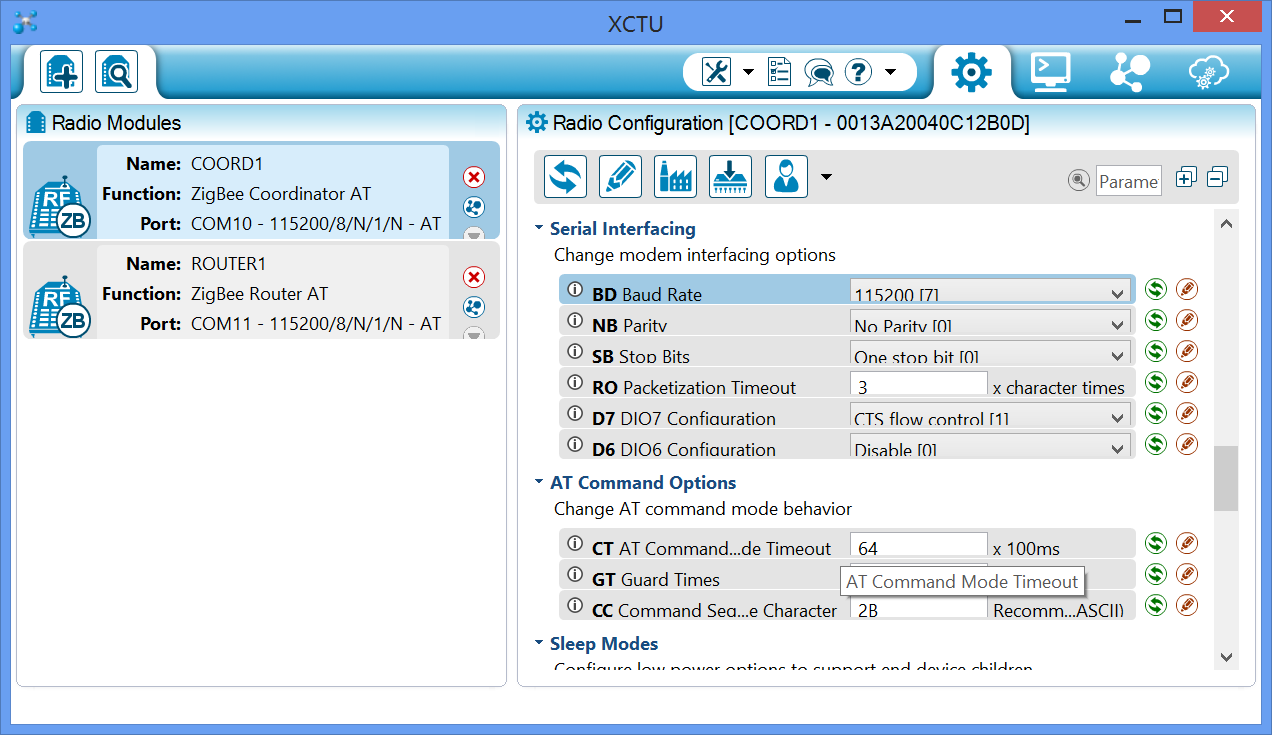
1. Next, we need to change some of the parameters on the ZigBee Router AT module, as shown in Figure 55. For this module, set the PAN ID (ID) to the same value that you used for the Coordinator (i.e. 2121 in this case).
2. Next, change the Destination High and Low Address (DH and DL) to be the Serial Number High and Low (SH and SL) value of the ZigBee Coordinator AT. As with the Coordinator you can change the Node Identifier (NI) to be a unique value, but this is not required.

  
Figure 55 - XCTU ZigBee Router AT Settings

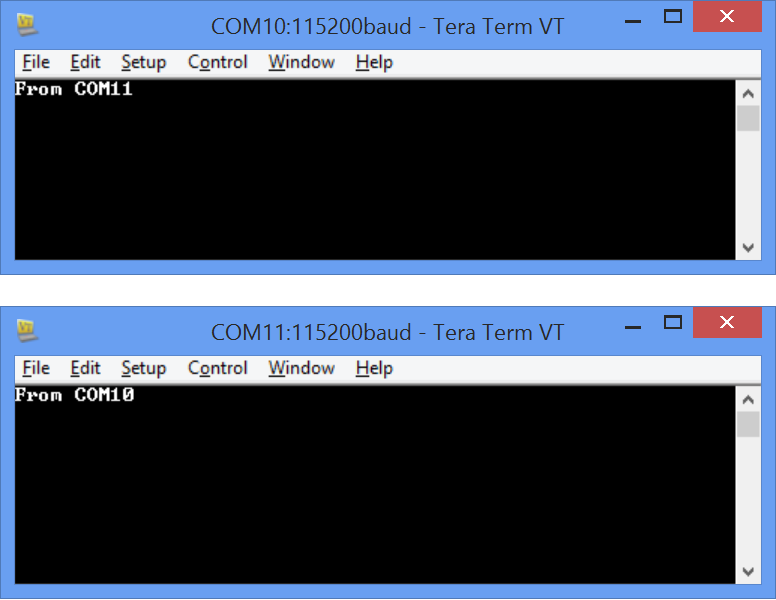
1. Finally, our Arduino sketch is using a baud rate of 115.2 kbaud, so we need to change the Baud Rate (BD) setting of both modules to be 115200.

This setting is a little over half way down in the list of settings for each device. Figure 56 shows the correct setting.

Once you have made all of the changes and written the values to flash, your modules should be setup to communicate with each other and act as the wireless serial adapter we require.

  
Figure 56 - XCTU Baud Rate setting for Both Modules

1. Next, you can use the terminal utility with [XCTU](http://www.digi.com/support/kbase/kbaseresultdetl?id=2125) or any [serial port terminal application](http://helpdeskgeek.com/windows-7/windows-7-hyperterminal/) to verify that your modules have been configured correctly. Figure 57 shows [Tera Term](http://ttssh2.sourceforge.jp/index.html.en) was used to connect both modules and send a message from each module to the other... Success!

  
Figure 57 - Testing the XBee Modules After Configuring

1. Next, unplug and remove one of the modules from the USB-to-XBee adapters (shown in Figure 46)<link to open in a new window: <fn> = fig46\_XbeeUsbAdapters.png, image located here: S:\Marketing Communications\TECHNICAL MARKETING\Content Team\Mouser OSHW Projects\Landa\Robotics - Project 1\images.zip> and place it in the socket on the Main System PCB. We will leave the other modules on the USB to XBee adapter for now to allow the Microsoft Windows' demonstration application to receive the information from the system. We are now ready to test so... put the batteries in and throw the lid on!

# Back to Top