



RadioBlocks Quick Start Guide

Contents

Thank you!	3
RadioBlocks, USB2UART and JTAG/ISP Boards	3
J2 Pin Out	3
Default Setup:	6
Python Tools Setup:	7
Default UART Settings:	7
Python Commander GUI:	8
Handy Test Application	11
Python Command Line Interface	13
Final Thoughts	15

Thank you!

Thanks for buying a RadioBlocks kit, we value and appreciate your business! For complete details on SimpleMesh or the Serial Protocol, please go to:

<http://www.coloradomicrodevices.com/software/documents/>

RadioBlocks, USB2UART and JTAG/ISP Boards

There are several kit variations but all kits have either, or both, a mains and/or battery powered RadioBlock. Battery powered nodes have a coin cell battery clip on back and the primary interface (J2) has female headers. The mains powered node does not have a battery clip and it has male pins on the primary interface (J2). The mains powered node and battery-powered node are shown in figure 1 and 2 respectively.

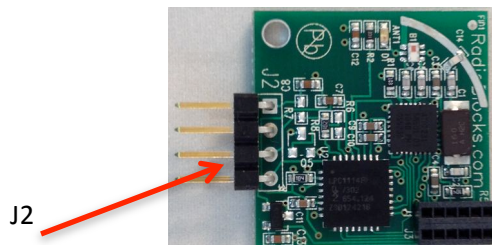


Figure 1 – Mains Powered

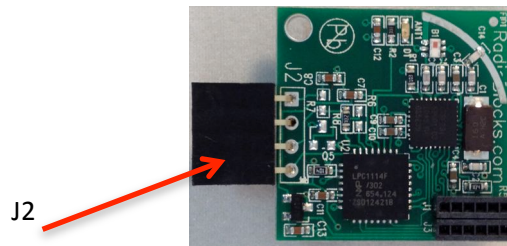
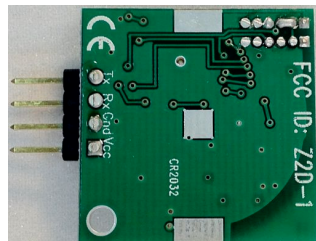
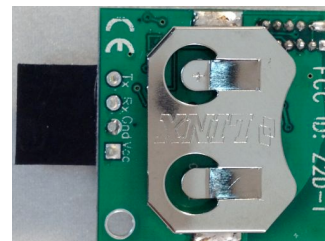


Figure 2 – Battery Powered

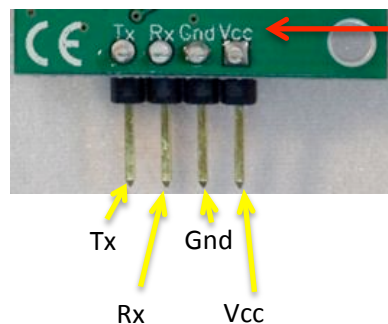


Mains Back



Battery Back

J2 Pin Out



Note that the pin out for J2 is labeled on the back of each RadioBlock

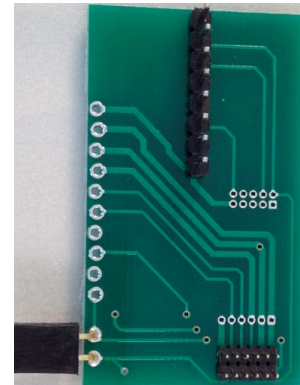
Other kits may have our USB2UART board and/or a JTAG/ISP programming board. The USB2UART has a CP2102 on board that converts USB signals to RS232 (TX and RX) signals and vice versa. The interface to a USB host is via its mini USB connector and the interface to the RadioBlock is via a 4-pin header. Note that Colorado Micro Devices supplies a 4-pin male/male connector kit to allow interfacing to battery-powered nodes, which have female headers. See figures 3 and 4.



Figure 3 – USB2UART



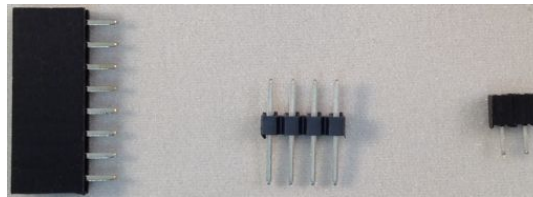
Figure 4 – JTAG/ISP



JTAG/ISP Back



JTAG/ISP Connector Kit



LPC Link – Battery Board USB – Battery Board ISP Connectors

The JTAG/ISP programming board is delivered configured for JTAG programming and debugging. It allows the RadioBlock to be connected via its J1 and J3 female headers to the male header pins on board the JTAG/ISP programming board. See figure 7.

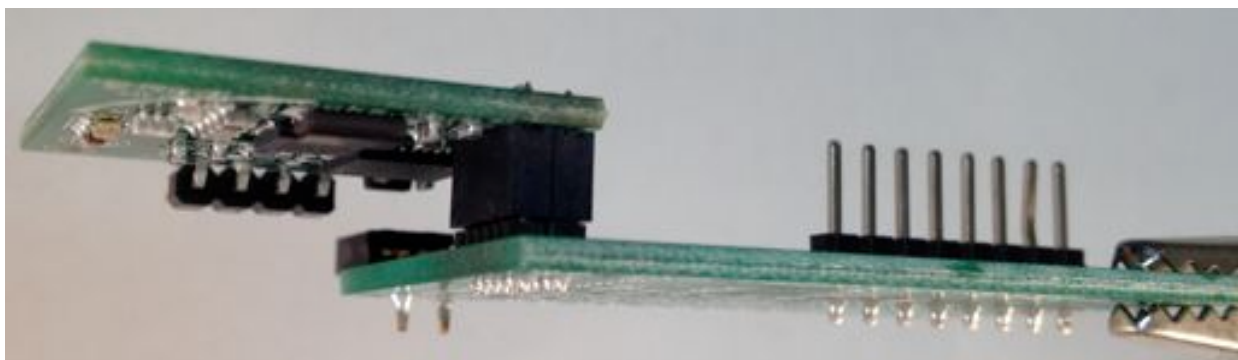


Figure 7 – RadioBlock connected to the JTAG/ISP board

The JTAG/ISP programming board can then be connected to the "LPC Link" JTAG board. You will need to purchase an LPC Link board from NXP. For more information see: <http://ics.nxp.com/lpcpresso/>. Note

that you get both the LPC Link board and the Code Red IDE, all for under \$30! To use the LPC Link board, you'll need to cut the traces between the vias (or separate the two halves). Then you can solder in the header we provide in the connector kit that come with the JTAG/ISP programming board. See figure 8.

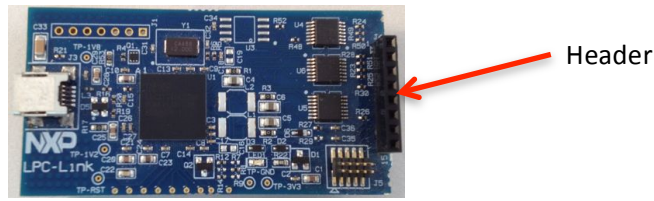
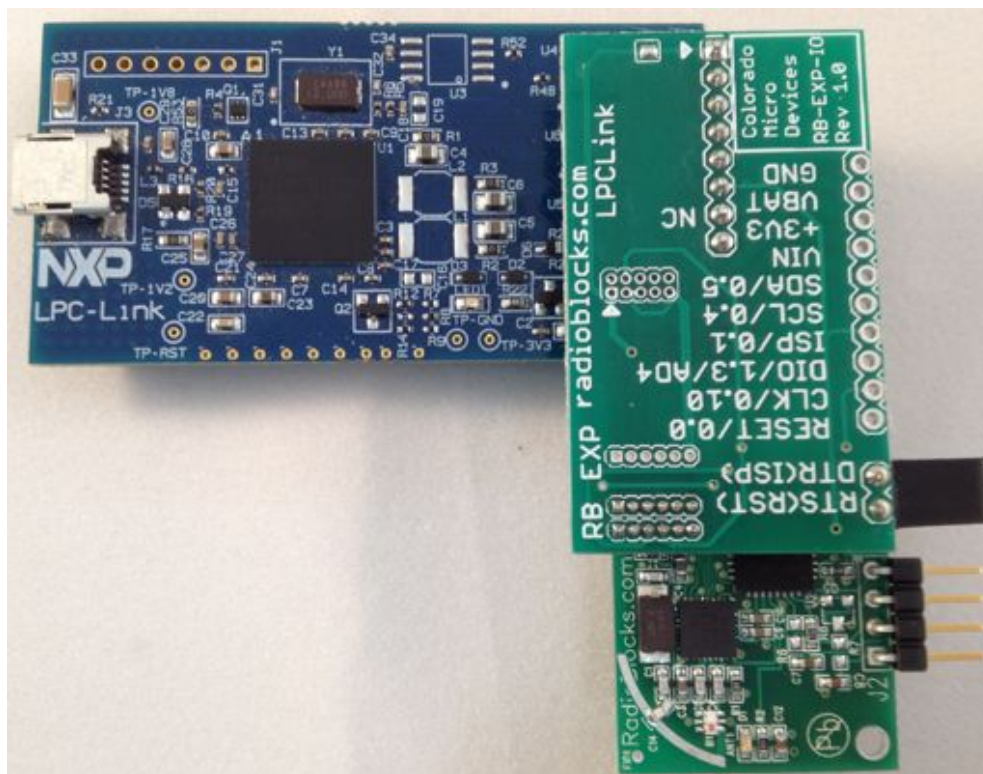


Figure 8 – LPC Link Board With header soldered in

Here's what the completely assembled RadioBlocks + JTAG/ISP board + LPC Link looks like when fully assembled:



RadioBlock Connected to LPC Link via JTAG/ISP Board

Default Setup:

All RadioBlock boards are delivered pre-programmed with SimpleMesh software loaded and the following addresses (at a minimum):

1. Mains powered node address = 0x0000
2. Battery powered node address = 0x0001
3. PanId = 0xABCD
4. Security key = "SimpleMesh_12345"

Some of the very first kits delivered did not have addresses indicated on a sticker on the back. Nevertheless those kits only had either one or two nodes and they conform to the address information provided above. Newer kits have the addresses on a sticker on the back of the board and the PanId and security data are as described above.

Python Tools Setup:

The Python Commander GUI (and the Python Sniffer GUI) requires three Python tools to be installed:

1. Python 2.7.x (<http://releases.qt-project.org/pyside/PySide-1.1.2.win32-py2.7.exe>)
2. PySide (http://qt-project.org/wiki/PySide_Binaries_Windows - Get the py2.7.exe version)
3. PySerial (<http://pypi.python.org/pypi/pyserial>)

Install Python first then the other two packages. Accept the defaults for all the installations.

Finally, you might have to add the path to the Python2.7.x executable to your PATH variable for your system. We presume you know how to do that...

Default UART Settings:

The default configuration of UART parameters is:

1. BAUD: 115,200
2. 8 Data bits
3. No parity
4. One Stop bit

Python Commander GUI:

Some notes about our Python Commander GUI. The Commander will connect to a RadioBlock using the USB2UART board. When the RadioBlock + USB2UART boards (figure 9) are connected via USB mini cable to your PC, launch the Python Commander GUI by typing, for example, "python rb_gui.py" at a command prompt or in your terminal. This assumes you have setup and installed the Python tools described earlier and cloned the "radioblocks- commander" from our open source GIT repository at: <http://www.assembla.com/code/radioblocks-commander/git/nodes>

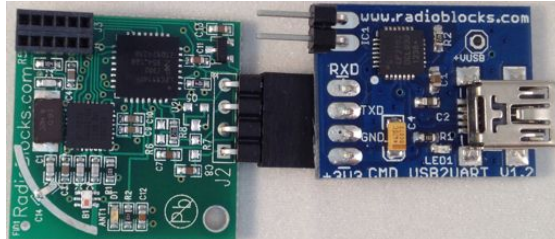
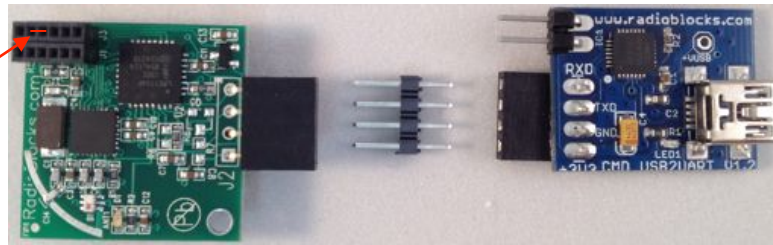


Figure 9 - Mains Powered Board Connected

Jump these pins to supply power on battery board.



Battery Powered Board Connection

Note:

Using the battery-powered board in this configuration will require a battery to be inserted or jumping pins 2 and 3 together on J3 to supply power to the RadioBlock.

Note:

The first time you plug in the USB2UART board it will locate and install the proper drivers. Let it do this automatically for you. For Windows users download from here

<http://www.silabs.com/products/mcu/Pages/USBtoUARTBridgeVCPDrivers.aspx>

The Commander GUI will start and you should see something that looks like the image in figure 10.

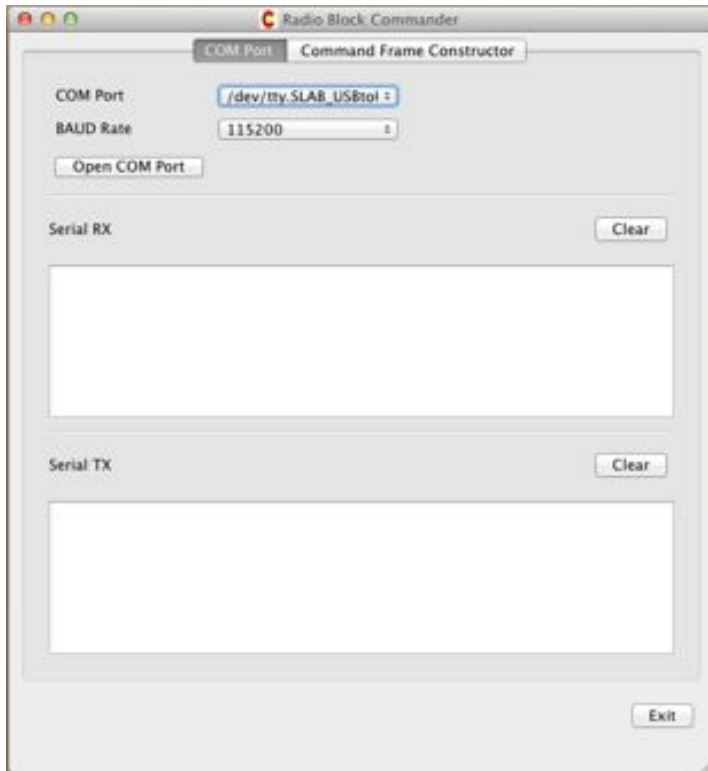


Figure 10 - Python Commander GUI

You need to first select the COM (or tty) port you have connected your USB2UART to. The Commander will populate the COM interfaces that it finds. We presume you know how to figure out which COM port the USB2UART board is connected to... Select the COM port from the drop down and the BAUD rate from the drop down. Recall the default BAUD rate is 115,200.

In figure 10, it can be seen that the proper COM port (/dev/tty.SLABtoUART on Mac/Linux or something like COM4 on Windows) and BAUD rate (115200) have been selected from the drop downs.

Next, select the "Command Frame Constructor" tab. The default serial command "test_request" should be highlighted, if you simply click the "Send Frame" button the Commander will formulate the test command and send it to the RadioBlock and display the results. You should see something similar to what is shown in figure 11 and if you do then you are connected! If not:

- Make sure you selected the proper COM port
- Quit the commander and unplug the device, plug it in again and try again
- Ensure that the SiLabs CP2102 driver got installed properly



Figure 11 - Successful Test Frame

Once you are connected and the RadioBlock is functioning you can use the Commander to send any command possible down the RadioBlock. See our "SimpleMesh Serial Protocol" document at: <http://www.coloradomicrodevices.com/software/documents/> for more information. Open the Command Selection drop down and scroll to the very last command "set_led_state_request", a second drop down will open and if you select "toggle" you can repeatedly turn the LED on and off by clicking on the "Send Frame" button.

Handy Test Application

If you have two RadioBlocks, a mains powered node connected to the Commander GUI via the USB2UART board and another mains powered node with a USB2UART, or a battery powered node, you can open the "send_data_request" command by clicking on it in the Command Selection drop down. To send the command to wirelessly turn on the LED on the battery node type in the address of the battery node "0x5555", for example, and then type "O" (capital O) in the Payload box then press return. Note that you must press return in the payload box after typing "O" (or whatever it is you are sending), this fires the appropriate signal/slot transaction in the underlying PySide (PyQt) code to populate the rest of the frame. This will load the command and calculate the CRC, the HEX command is shown in the "Serial frame to send" box. If you then click on the Send Frame button the frame will be transmitted wirelessly to the battery node and its LED should come on! You should also see a response in the "RadioBlock Response" window with an appropriate message, see figure 12.

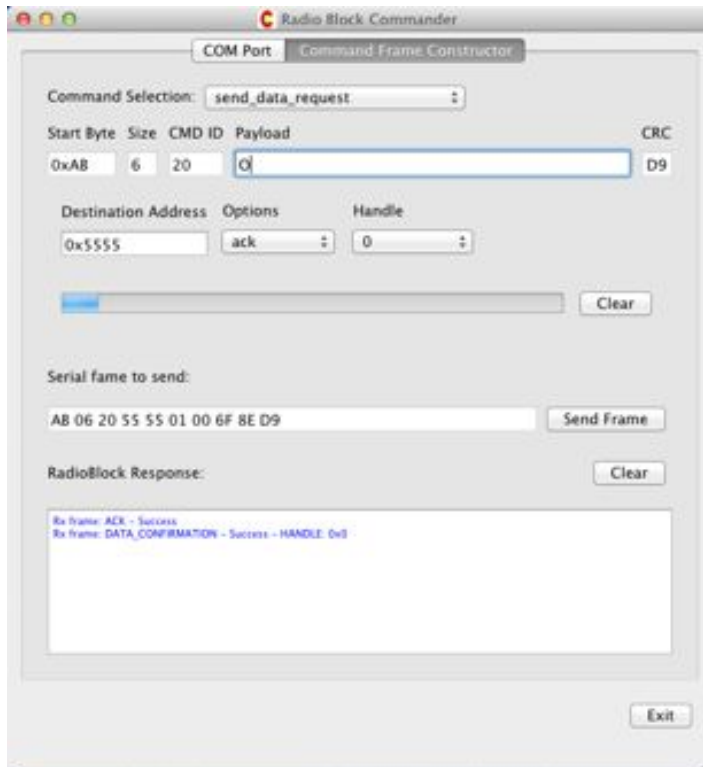


Figure 12 – Send Data Request Example

If you had a mains powered RadioBlock and another USB2UART connected that received the wireless frame, and you were running another instance of RadioBlock Commander, you would see this on the receiver:

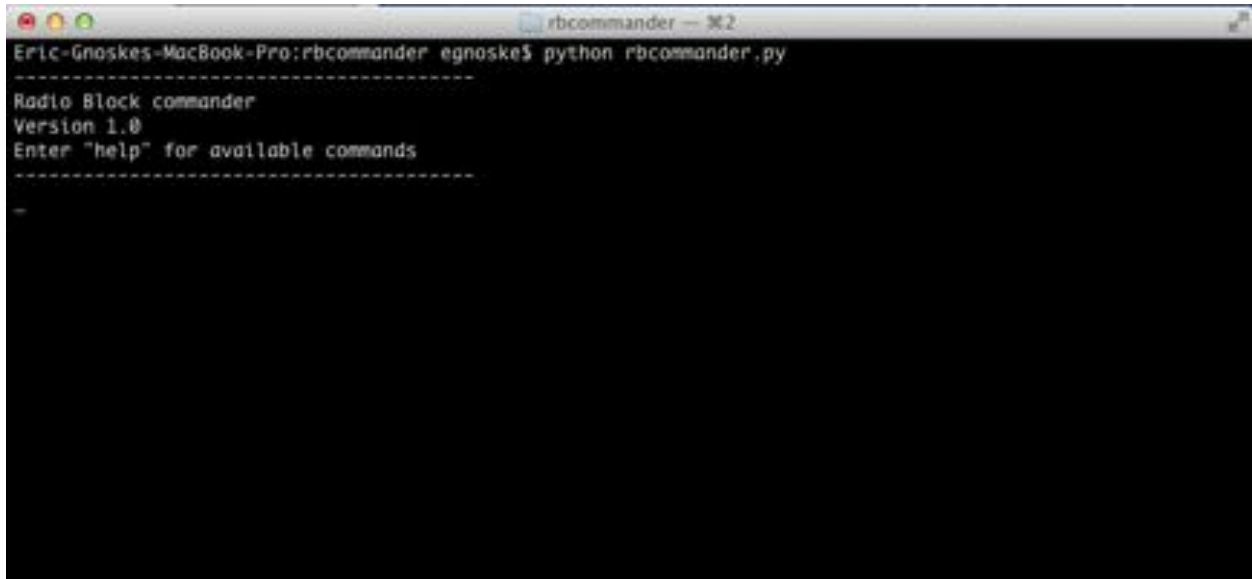


Figure 13 - Received Data

Where the received byte, 0x6F, is ASCII for "O" – Woo Hoo, Over The Air fun!

Python Command Line Interface (CLI)

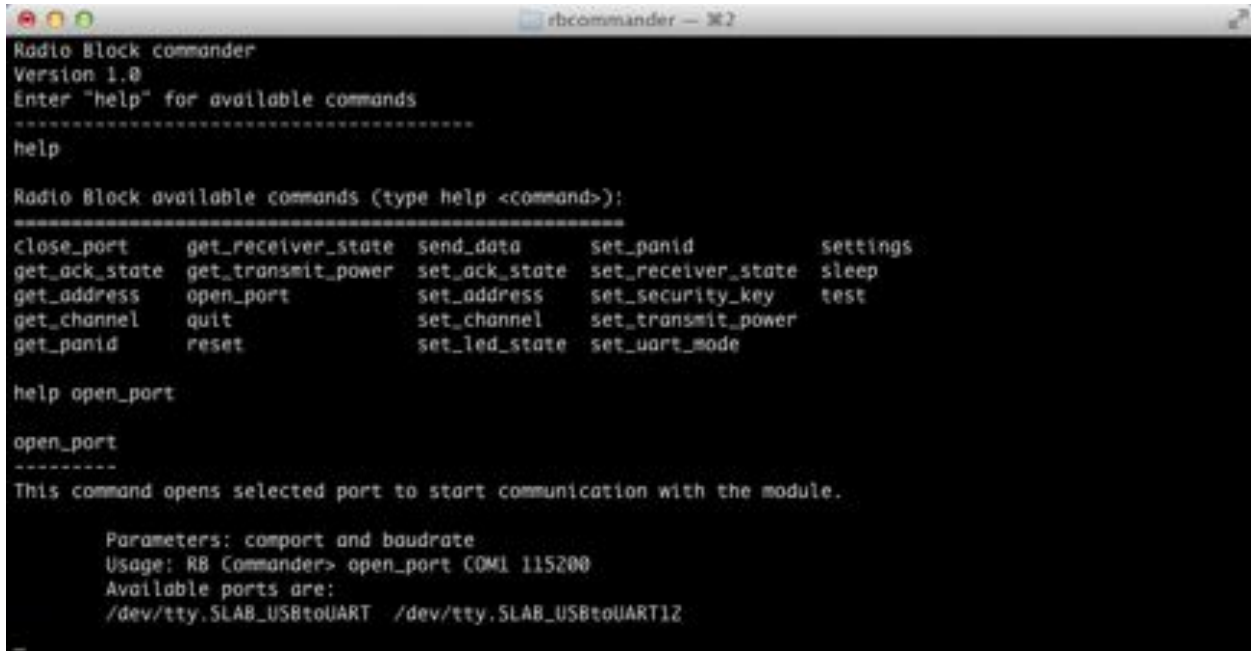
Some folks want to be able to script commands or just prefer operating from the command line. No worries, we've provided you with a command line interface as well. Go to a command line and navigate to the "radioblocks-commander" directory and run this, for example, `C:\work\radioblocks-commander\python rb_cli.py` – You will see the something like Figure 14.



```
Eric-Gnoskes-MacBook-Pro:rbcommander egnoske$ python rbcommander.py
-----
Radio Block commander
Version 1.0
Enter "help" for available commands
-----
-
```

Figure 14 - CLI, rb_cli.py

Now type *help* to see a list of commands and you should see the commands and if you type *help command* you'll see help for that command. See my example in Figure 15 where I got help on the *open_port* command which also returned the available ports that had USB2UART device plugged in!



```

Radio Block commander
Version 1.0
Enter "help" for available commands
-----
help

Radio Block available commands (type help <command>):
-----
close_port      get_receiver_state  send_data        set_panid        settings
get_ack_state   get_transmit_power  set_ack_state    set_receiver_state  sleep
get_address     open_port           set_address      set_security_key   test
get_channel     quit                set_channel      set_transmit_power
get_panid       reset               set_led_state    set_uart_mode

help open_port

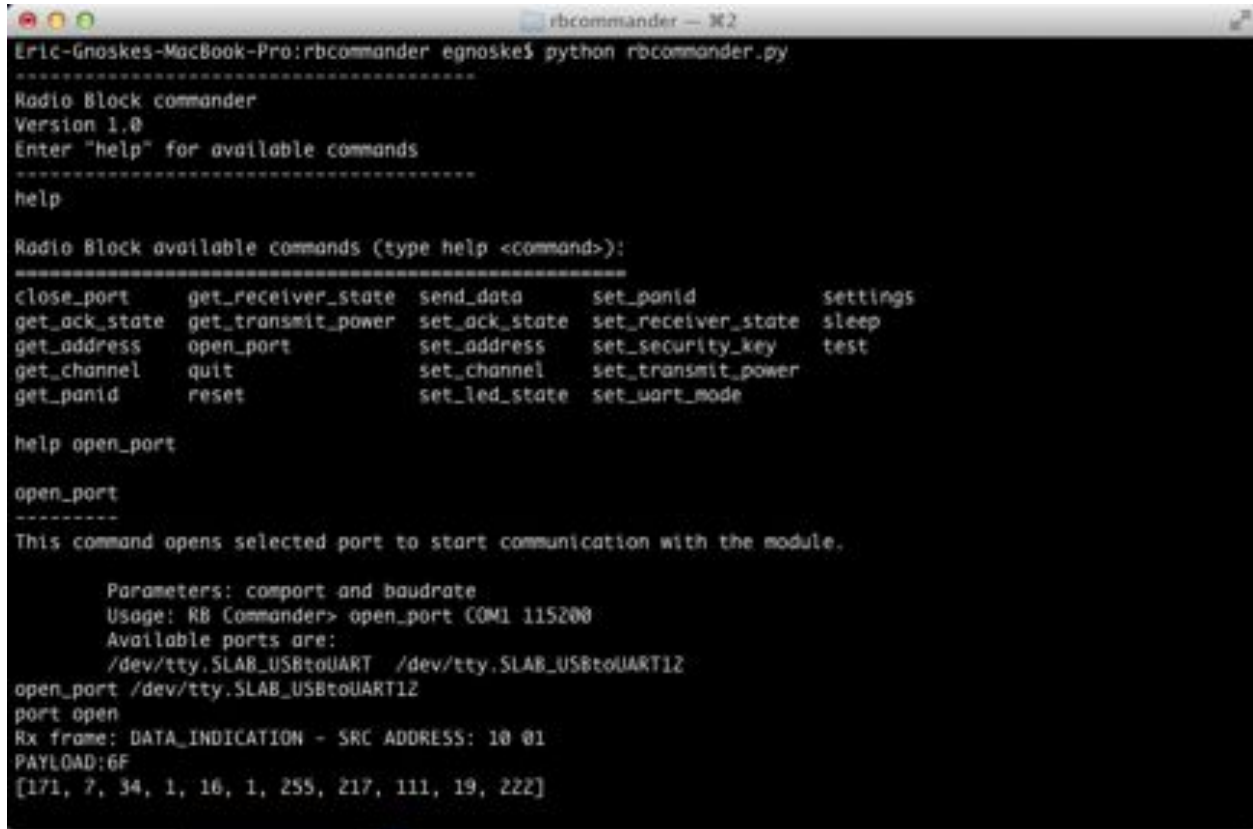
open_port
-----
This command opens selected port to start communication with the module.

Parameters: comport and baudrate
Usage: RB Commander> open_port COM1 115200
Available ports are:
/dev/tty.SLAB_USBtoUART /dev/tty.SLAB_USBtoUART12
  
```

Figure 15 - Help with *open_port* Command

Since each command has help associated with it we hope you can play around with it to see how it all works. We think it has been made “Simply Useful!”

One last show, we show what happens when this node receives the OTA led on command from a transmitter node – Identical to the GUI example above. See Figure 16 below:



```

Eric-Gnoskes-MacBook-Pro:rbcommander egnoske$ python rbcommander.py
.....
Radio Block commander
Version 1.0
Enter "help" for available commands
.....
help
.....
Radio Block available commands (type help <command>):
.....
close_port      get_receiver_state  send_data      set_panid      settings
get_ack_state   get_transmit_power  set_ack_state  set_receiver_state  sleep
get_address     open_port           set_address    set_security_key  test
get_channel     quit                set_channel    set_transmit_power
get_panid       reset               set_led_state  set_uart_mode

help open_port

open_port
-----
This command opens selected port to start communication with the module.

Parameters: comport and baudrate
Usage: RB Commander> open_port COM1 115200
Available ports are:
/dev/tty.SLAB_USBtoUART /dev/tty.SLAB_USBtoUART12
open_port /dev/tty.SLAB_USBtoUART12
port open
Rx frame: DATA_INDICATION - SRC ADDRESS: 10 01
PAYLOAD:6F
[171, 7, 34, 1, 16, 1, 255, 217, 111, 19, 222]

```

Figure 6 - Data Indication Showing "O" Received Over The Air (OTA)

Final Thoughts

All this software and, indeed, the firmware (SimpleMesh) has been written by several folks and contributed as *open source* code to support RadioBlocks. We hope there are folks out there who will want to help out and make the code, documents and overall RadioBlocks experience better. Please visit our forum at <http://www.coloradomicrodevices.com/forums/forum/colorado-micro-devices-forum/> and post corrections, comments and wisdom - Thanks!!