USERS GUIDE

EK2100 Evaluation Kit

for SNAP and Portal Version 2.2

Document Revision v1.2

Wireless Technology to Control and Monitor Anything from Anywhere™

132 Export Circle
Huntsville, Alabama 35806
877-982-7888

Doc# 600-0014C

Table of Contents

1. Before Getting Started

Welcome to the *EK2100 Users Guide*. This document is a quick tutorial introducing you to the Synapse SNAP product line using a couple of simple, hands-on demonstrations.

This manual is intended to be a brief introduction and includes only the information needed to run the evaluation and sample applications. Much more information is available via other manuals and tutorials. It also focuses on the components actually included in the kit, rather than trying to cover all the different types of SNAP hardware that are available. Just be aware that there exist other types of SNAP-compatible hardware than what you see included in this kit.

Because it is intended to be a **tutorial**, you need to read through it *in order*, as opposed to skipping around within the document. You also need to *actually do the steps* as specified, because later sections assume the steps from previous sections have been completed.

Other Documentation

This document, the EK2100 Evaluation Kit Users Guide, is only one of several included on the CD that comes with the EK2500 kit. Be sure to also take a look at:

- The "SNAP Reference Manual" (600-0007B)

  This document contains lots of information on how to use Portal, the software that runs on your PC and allows you to configure and manage your wireless network. This document is also where you will find information on the built-in functions provided by Portal and SNAPpy.

- The "SNAP Hardware Technical Manual" (600-101.01C)

  Every switch, button, and jumper of every SNAP board is covered in this hardware reference document.

- The "End Device Quick Start Guide" (600-0001A)
- The "SN171 Proto Board Quick Start Guide" (600-0011C)

  These two documents are subsets of the "SNAP Hardware Technical Manual", and come in handy because they focus on a single board type.

- The "SNAP Sniffer Users Guide" (600026-01A)

Starting with Portal version 2.2.23, a "wireless sniffer" capability is included with Portal. If you follow the instructions in this standalone manual, you will be able to actually *see* the wireless exchanges that are taking place between your SNAP nodes.

For version 2.2, we are going to be splitting the existing "SNAP Reference Manual" up into multiple smaller documents. The new documents include:

- The "Portal Reference Manual" (600024-01A)

  Until this document comes out, Portal will be covered in the "SNAP Reference Manual". This new manual will cover all of the features of the Portal GUI in a standalone document.

- The "SNAP Users Guide" (600025-01A)

  This new document will bridge the gap between the various "kit" User Guides and the "SNAP Reference Manual". It will cover the basics of SNAPpy programming.

- The "SNAP 2.2 Migration Guide" (600023-01A)

  There were enough changes between the 2.1 and 2.2 series of SNAP releases that we decided to provide an extra "transition" guide. You should check this document out if you were already a user of SNAP 2.1 and Portal 2.1.

All of these documents are in Portable Document Format (PDF) files on the CD included with each evaluation kit.

Other Sources of Information

There is a dedicated support forum at http://forums.synapse-wireless.com.

In the forum, you can see questions and answers posted by other users, as well as post your own questions. The forum also has examples and Application Notes, waiting to be downloaded.

You could also check the company website at www.synapse-wireless.com.

2. Getting Started

Overview

The Synapse ***SNAP*** product family provides an extremely powerful and flexible platform for developing, deploying and managing 802.15.4 based wireless applications.

SNAP stands for Synapse Network Appliance Protocol. The term is also used somewhat generically to refer to the entire product line. Often when we are talking about SNAP we are implicitly including SNAP, SNAPpy, Portal, and SNAPconnect.

A *SNAP* network consists of individual SNAP nodes. At the heart of each node is a Synapse RF Engine.

Each RF Engine combines a microcontroller, an 802.15.4 radio, and an antenna. The antenna can either be an integral "F" antenna, or a mounting point for an external antenna.

Some RF Engines also include an internal transmit Power Amplifier (PA) for maximum radio range. These nodes can be identified by the wording "RFET" on their label. RF Engines without the extra Power Amplifier are labeled "RFE". All versions have a receive amplifier for extended range.

> ***Note:*** *The EK2100 evaluation kit only includes Synapse RFET engines (with an integrated F antenna and an internal PA). Other types of SNAP engines (both from Synapse or other vendors like Panasonic) can be purchased and are fully compatible with both the SN171 Proto board and the SNAPstick USB node. It is important to understand that any type of SNAP Engine can be used with any Synapse "demonstration" board; this is just the combination chosen for this kit.*

The on-board microcontroller has its own internal RAM and ROM. No external components are required for operation.

Each Synapse RF Engine includes 19 General Purpose I/O (GPIO) pins, which can be configured as digital inputs or outputs. Many of these same 19 GPIO pins can also be switched to alternate functionality:

- 8 can be analog inputs
- 4 can be serial data lines (2 TX pins, 2 RX pins)
- 4 can be serial handshake lines (2 RTS pins, 2 CTS pins)

Although 19 I/O pins are available, you only have to hook up the exact functionality required by your application. The *minimal* hookup to an RF Engine consists of two wires:

- One wire for VCC (2.7-3.4 volts DC)
- One wire for GND

The RF Engines contain core code (written in C) that implements basic wireless networking functionality. This core code also implements an interpreter for a subset of the Python programming language. Synapse has named this subset of Python **SNAPpy**.

> You can find details on the SNAPpy language, and how it compares to Python, in the SNAP Reference Manual. For now, the important point is to understand that SNAP nodes do implement a scripting language.

SNAPpy scripts can be uploaded into RF Engines "Over The AIR" (OTA), or over the serial interfaces. These scripts effectively define the personality of each node – by changing the SNAPpy scripts uploaded into a node, you can completely change the node's behavior.

RF Engines can be designed into your own products, acting as a slave device to your main microcontroller or microprocessor. In many cases the RF Engine can take over the functionality of the original main processor, in *addition* to adding wireless capability and SNAPpy scripting.

In addition to discrete RF Engine modules, Synapse also sells a growing portfolio of SNAP **nodes** that accept RF Engines, but then extend their capabilities with additional I/O hardware.

Two different kinds of SNAP nodes are included in this evaluation kit:

- One USB SNAPstick
- One SN171 Proto Board

Details about each of these boards can be found in the SNAP Hardware Technical Manual, as well as their associated 'quick-start' guides

A Portal into Your Network

Portal is a standalone software application which runs on any standard PC with Microsoft Windows 2000 or higher. Using a USB or RS232 interface, it connects to any ***RF Engine*** in the SNAP Wireless Network, becoming a user interface for the entire network.

Note: – most of the icons shown in the previous diagram were taken directly from the Portal user interface.

 is used within Portal to generically represent a SNAP node.

 is used within the Portal user interface to represent Portal itself.

The version of Portal that comes with this particular evaluation kit supports six nodes, but you can license versions for higher node counts.

Once connected, Portal provides the capability to *interactively* build an *intelligent* wireless network. You can:

- Discover new SNAP Devices
- Upload *intelligence* to those Devices over the air, using SNAPpy scripts
- Customize Portal to suit your specific application

Interactively - you do all this within Portal, observing the results immediately.

Intelligent - the network is purpose-built for your application.

Synapse's ***Portal*** administrative software is included in this evaluation kit. Details regarding ***Portal's*** installation and use are presented later in this document.

For more information about using Portal, refer to the "Portal Reference Manual".

Throughout this manual, the term "Portal PC" is used to refer to the PC that Portal is running on.

SNAPconnect

Synapse also licenses a standalone *SNAPconnect* application, giving you XML-RPC access into your SNAP network. Running on Windows or Linux, this software provides a XML-RPC API which can be used by client software written in Python, JAVA, C#, C++, etc. This allows backend systems to participate seamlessly in the SNAP network.

This *SNAPconnect* software is not included in this evaluation kit, and is not mentioned in the remainder of this document.

Evaluation Kit Hardware

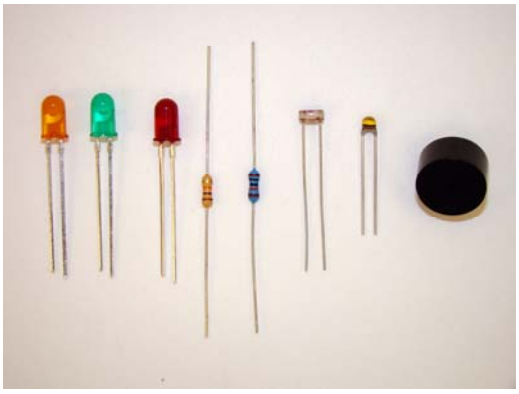
To demonstrate many of the capabilities of Synapse ***RF Engines***, ***SNAP*** and ***Portal***, we've bundled them together in an evaluation kit form – the EK2100.

The EK2100 evaluation kit comes with management software, a power supply, an external battery holder, a pair of AA batteries, a screwdriver and a bag of components (refer to the *EK2100 Kit Contents* document). The kit also contains *two different kinds* of SNAP demonstration nodes:

- One SN132 SNAPstick USB Module
- One SN171 Proto Board

> ***Remember****: You do not have to use the various "demo" boards in order to use the RF Engines in your own projects.*
>
> ***Also:*** *If you crave even more fun, additional boards can be purchased from Synapse to expand you evaluation network.*

We tend to hate to hear the words "*some assembly required*". Hopefully, we can make an exception with this kit. As you work through the upcoming demonstrations you will get the chance to "play" along and connect various sensors and indicators. Your kit should have come with a screwdriver and bag of various electronic components.



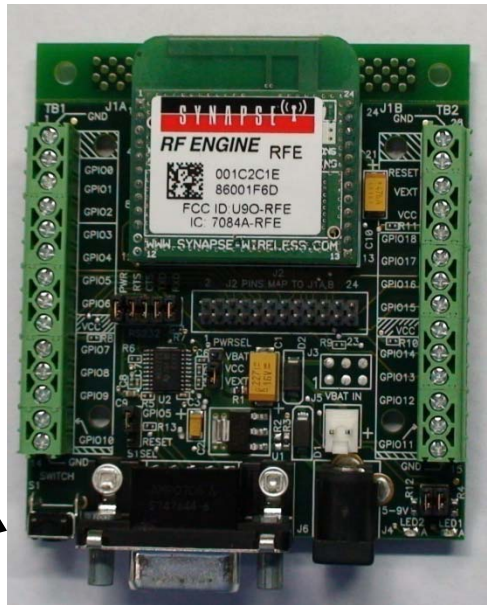
Your bag should include the following:

- 3 LEDs (1 amber, 1 green, 1 red)
- 1 10K Ohm resistor (beige)
- 1 100K Ohm resistor (blue)
- 1 Photo-cell (circular sensor)
- 1 Thermistor
- 1 Piezo buzzer (small black disc)

Before getting down to business let's talk a little more about the two SNAP demonstration boards included with this kit:

SN171 Proto Board

Select or Reset Button



The Synapse SN171 Proto (prototyping) Board provides easy access to all 19 General Purpose Input/Output (GPIO) pins available on the SNAP RF Engine. 8 of these pins (GPIO 11-18) can also serve as analog inputs.

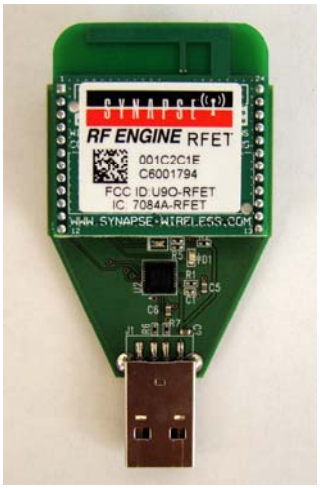
All of the onboard RF Engine (RFE) signals are made available at easy access screw terminal blocks located on either side of the node. These same signals are also made available in a more compact form in the center of the board. A single dual row header provides a connection point for a ribbon-cable (or some other form of wiring harness) from some other circuit board.

The SN171 can be powered by an external power supply (5-9 VDC) or two AA batteries. An external battery holder is also included in the kit for battery operation.

One quick note: The SN171 does not support the same external voltage range as other Synapse evaluation boards. It supports a range of only 5-9 VDC.

More information about this module's hardware can be found in the *SN171 Quick Start Guide* and the *SNAP Hardware Technical Manual* found on the included CD.

SN132 SNAPstick USB module

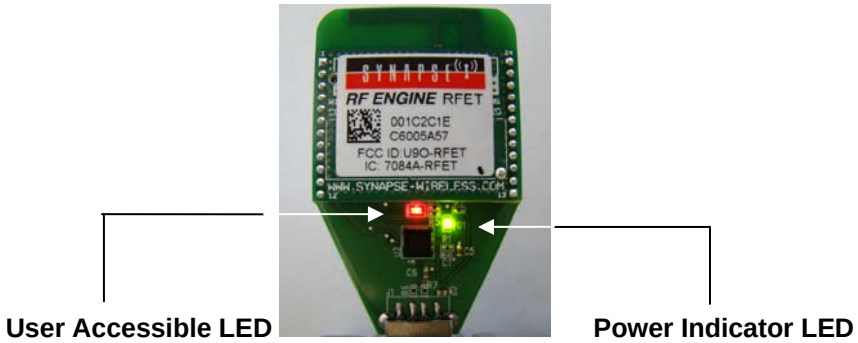


The Synapse SN132 SNAPstick USB module does not provide access to the entire range of GPIO pins available through the SN171 Proto Board. Instead, it provides a simple and compact way to connect a PC running Portal to a SNAP network.

A Tri-color LED is available as an output indicator. This component has the ability to emit a red, green, or amber light. It can be controlled by manipulating GPIO pins 0 and 1 via SNAPpy scripts. The following table describes the interaction of output pins and resulting colors. Notice that the LED lines are active LOW.

Desired LED Color	*Value of GPIO Pin 0*	*Value of GPIO Pin 1*
Red	Low	High
Green	High	Low
Amber	Low	Low

A second green LED is used to indicate that power is being supplied to the module. It cannot be controlled by the user.



Power to this module is provided via a standard USB connection. This allows for the SNAPstick to be powered using a PC or other USB power source (such as an AC adaptor).

> ***Note:*** *The Portal software does not need to be installed for the SNAPstick to draw power from a PC's USB port.*

More information about this module's hardware can be found in the *SNAPstick Quick Start Guide* and the *SNAP Hardware Technical Manual* (located on the included CD).

Demonstration Time

The following section takes the user through a series of step-by-step demonstrations. These demos will serve as a brief introduction to the capabilities of Synapse's SNAP RF engines and Portal software.

Each segment builds upon the concepts of the previous demonstration and they are most effective if executed in order. However, this is certainly not mandatory.

SNAPstick USB Connection

Go ahead and connect the SNAPstick USB module to an open USB slot on the PC or a USB power adapter. If you do connect to a PC, the following dialog box will appear:

Since we have not installed the correct software yet (we'll do that later in this document), we are not ready to do this - just click on "**Cancel**" for now.

Demonstration 1: Counting Straight Out of the Box

Ok. Now it's time to actually play with our new toys.

Both nodes come from the factory preloaded with a demo SNAPpy script. Although this demo script showcases only a small portion of the capabilities of a SNAP node, it provides a quick way to get familiar with the nodes, without having to install any software on your PC.

GOAL: To demonstrate how SNAP nodes can interact without the need for controller software. They work straight out of the box.

Step 1 – By now you should have the SNAPstick USB module plugged into your PC or USB power supply. The module will have a single green LED lit to indicate it is up and running.

Try and keep the SNAPstick in a place where you can see the LEDs. Since this is the only form of output on this particular node, we'll use them in a couple of demonstrations.

> ***Note: Don't install the software just yet.***
> *While the module is drawing power from the USB connection, it is not yet interacting with any installed software. SNAP RF engines (and the demonstration boards that utilize them) do not require the Portal software to operate. Once a SNAPpy script is loaded into a unit, it can become a completely autonomous node in a network.*

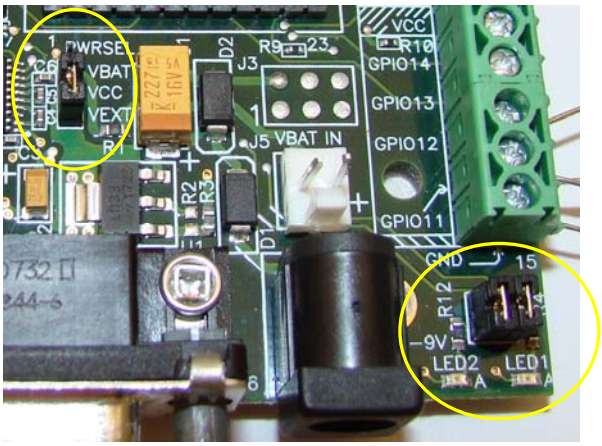
Step 2 - **Power up** the SN171 proto-board using the 9V AC adapter provided with your evaluation kit. You should see the yellow LED (LED2) begin to flash. This means we are ready to push some buttons.

If your nodes are on and each has a lit LED, proceed to the next section. Otherwise, here are some trouble-shooting tips that may help:

Troubleshooting

Tip #1: Verify the power related jumpers*:*
Your SN171 "proto-board" node should have come from the factory preconfigured to work with an external DC power supply. Still, as a double-check, verify that the VCC jumper is in the VEXT position (connecting pins 2-3), not the VBAT position (connecting pins 1-2).

Tip #2: Verify the LED related jumpers:
The SN171 "proto-board" node should come from the factory preconfigured to enable its two on-board LEDs (one yellow, one green). If the unit is not blinking its yellow LED, it is worth verifying that the LED1 and LED2 jumpers are both installed.

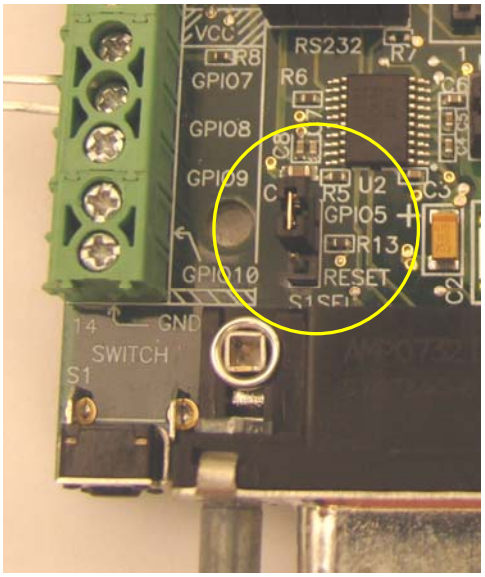


Now back to the action…
The pre-loaded script ("McastCounter.py") will keep track of a global count. This count is incremented every time a 'button press' is registered.

Step 3 - **Push** the button on the proto-board. You will notice the LEDs on the proto-board and SNAPstick change pattern with each button press. Press-and-hold the button and you will see the LED configuration return to its original state (i.e. the count returns to zero).

More Troubleshooting

If you are not seeing the LEDs change on the proto-board make sure that the button is enabled. The jumper located up and to the right of the button is set to for 'GPIO5' (connecting pins 1 and 2) and not to 'reset' (pins 2 and 3).



This demonstration is a simple way to show a couple of key points:

- The nodes are immediately able to communicate with each other - There is no such thing as a "network join time" with SNAP.

- Any node can talk to any other node - There is no central "coordinator" node with SNAP.

- No PC software required – There is no need for software installed on a PC to "coordinate" the nodes, they can think for themselves.

- Using SNAPpy scripts, SNAP nodes can autonomously respond to changes in their environment – This button press could easily be some form of digital or analog sensor reading.

The counting you just observed is accomplished using the multicast capabilities of SNAP. If other modules were added to the network, they too would track the button count and could even participate in the counting.

The first example was pretty simple. Let's expand on things and include a look at how nodes can interact with Synapse's Portal software.

So, without further ado, we will move onto the next step; installing Synapse's Portal software application…

3. Installing Portal

The Synapse Network Evaluation Kit contains a CD with the Portal Installer as well as RF Engine support documentation. This CD provides all the software you need to get started with SNAP and Portal. Additional updates and the most current release of this software will always be available on the Synapse support forum – so you may want to check for updates before installing.

> ***Note:*** *The version numbers related to the software you are installing may be higher than those depicted in the following screen-shots*

Updates on the Web

You'll find the latest version of Portal on the Synapse Support Forum at http://forums.synapse-wireless.com (look under Software Releases -> latest Releases)

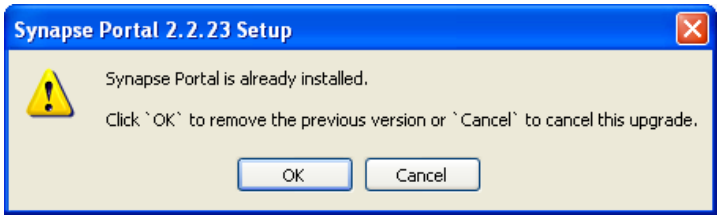
Portal comes bundled with the latest SNAP firmware and documentation.

There is also a company website at http://www.synapse-wireless.com

Running Setup

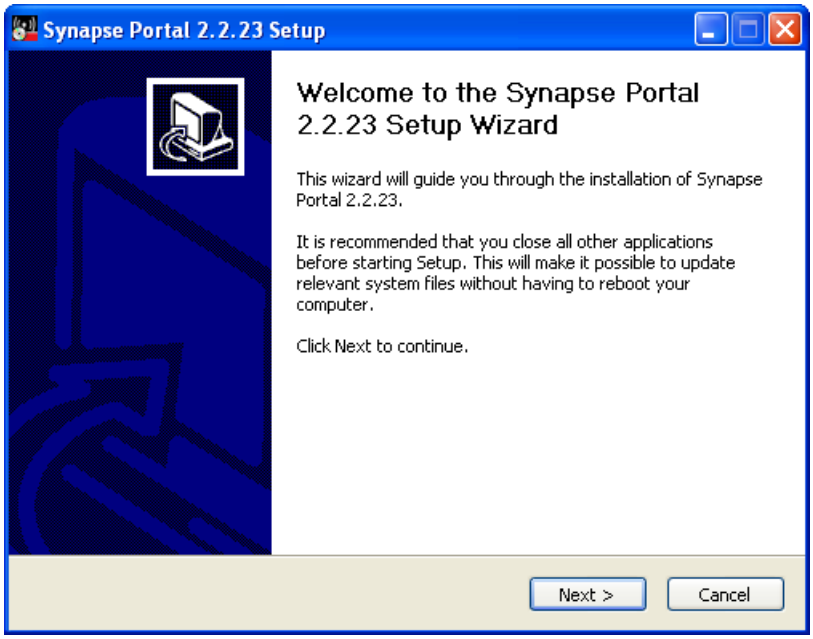
Insert the **Synapse Portal CD,** and run the **Portal -EK2100-setup.exe** you'll find there. (Windows "AutoRun" may launch this for you automatically)

NOTE – if a previous version of Portal (for example, version 2.0) was already installed on your computer, you will get an initial dialog box like the following:

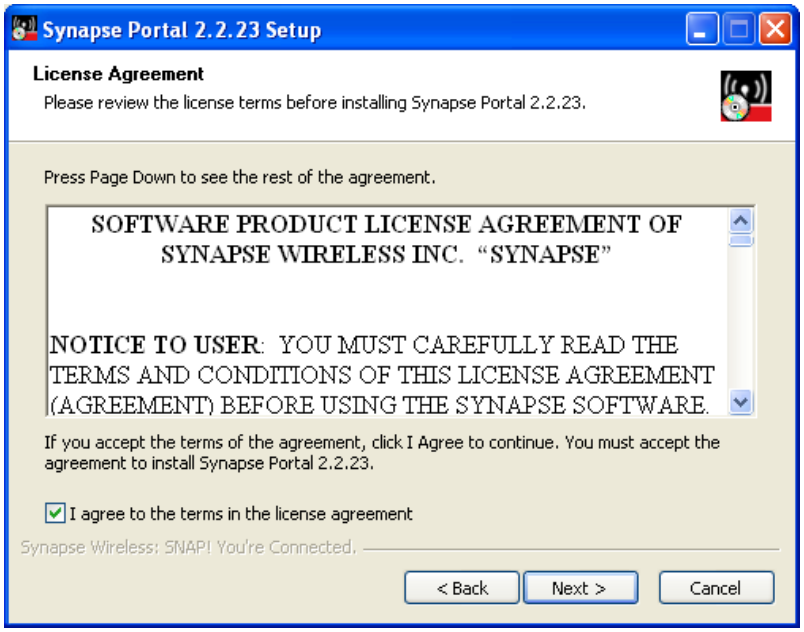


You must click the "OK" button in order to install the newer version.

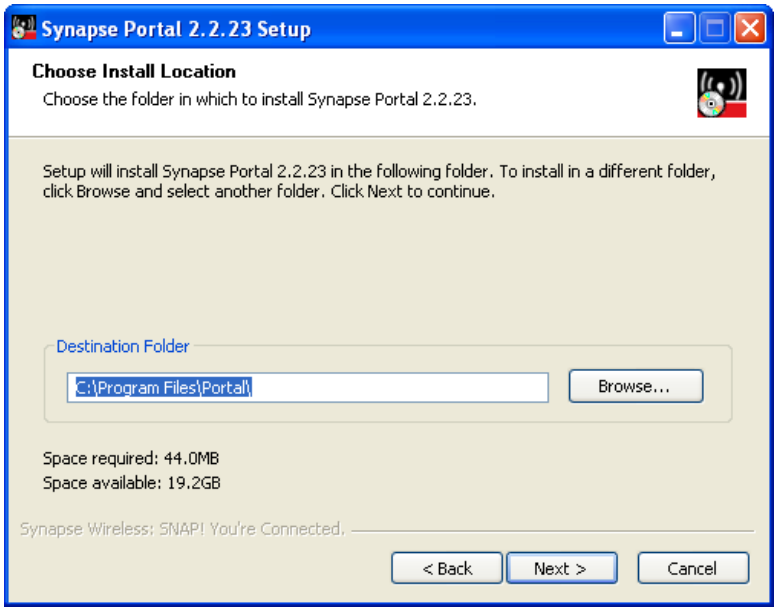
A dialog box similar to the following will appear (your version number may be higher).



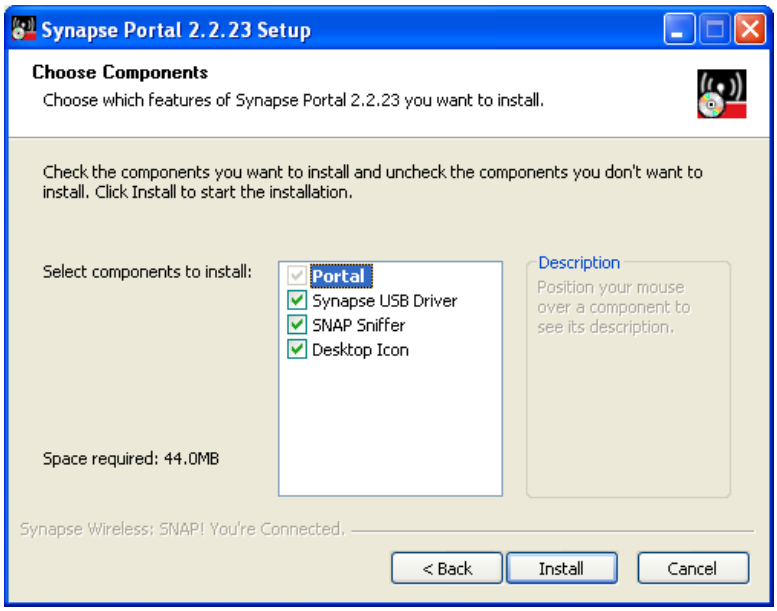
Click the **Next** button to get the following:



Read the license agreement, check the "***I agree***" box and then click on **Next**.

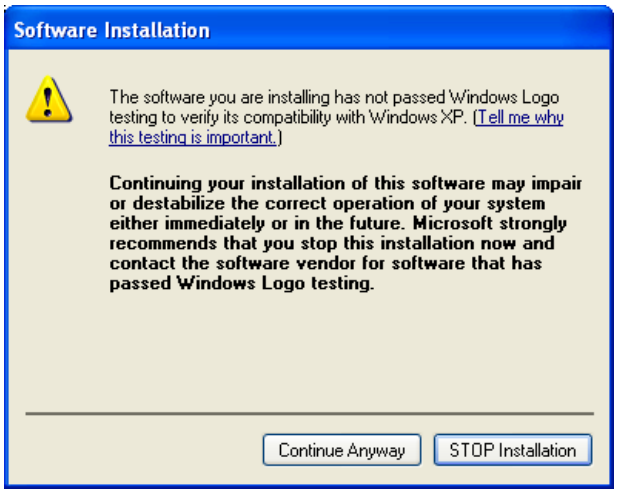


You can either enter the desired destination folder manually, browse to the desired folder, or just click on **Next** to accept the default.



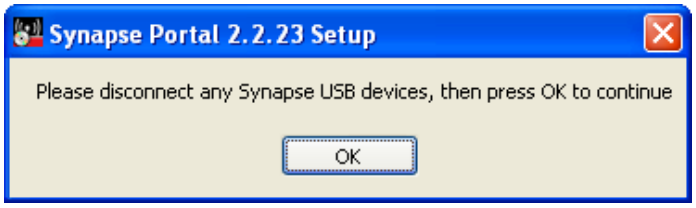
Make sure the desired components are checked, and click on **Install**.

Depending on the version of Windows running on your PC, you may get a warning dialog similar to the following:

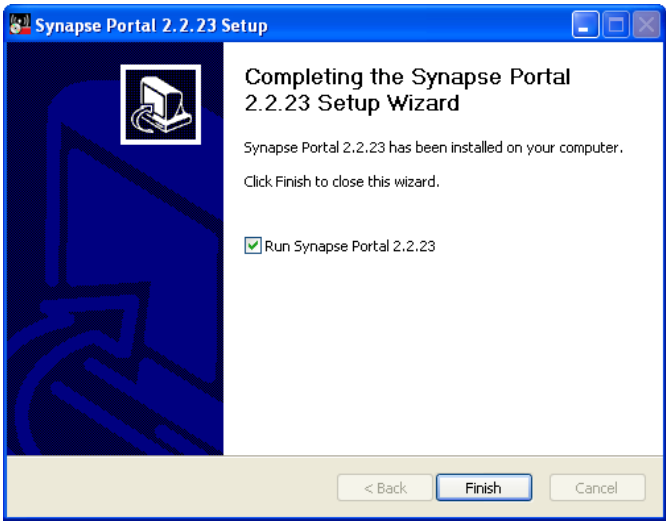


The warning is harmless, and you should click on ***Continue Anyway*** to proceed with the installation.

After several files have been processed, you will get the following dialog box:



To ensure that the latest Synapse USB drivers can be installed, you must not be running the old versions of these drivers. Disconnect any Synapse USB devices to ensure this, and then click on the "OK" button. The installation process will continue.



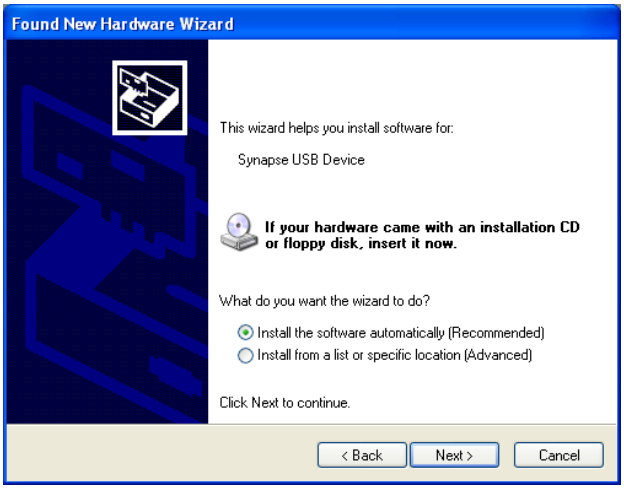
You have now successfully installed ***Portal***. There will be a ***Portal*** icon on your Windows *Desktop*, as well as in the *Start Menu*.

> ***Note:*** *The Synapse Portal CD also contains Hardware and Software Documentation and Release Notes.*

USB Setup

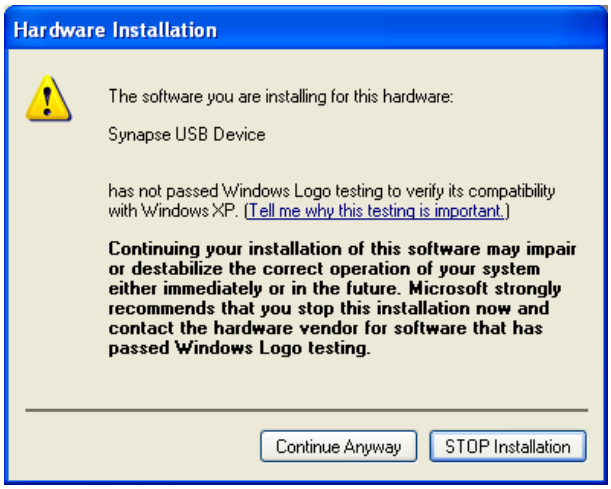
The SNAPstick requires that the Synapse USB drivers be installed in order to properly communicate with Portal. Make sure the SNAPstick module is connected to an available USB port. Since we have yet to install any drivers for this device, the following dialog box should appear:

The correct software is already available (you just installed it when you installed Portal!), there is no need for Windows to connect to Windows Update – just click on "**No, not this time**" and then click on **Next**.

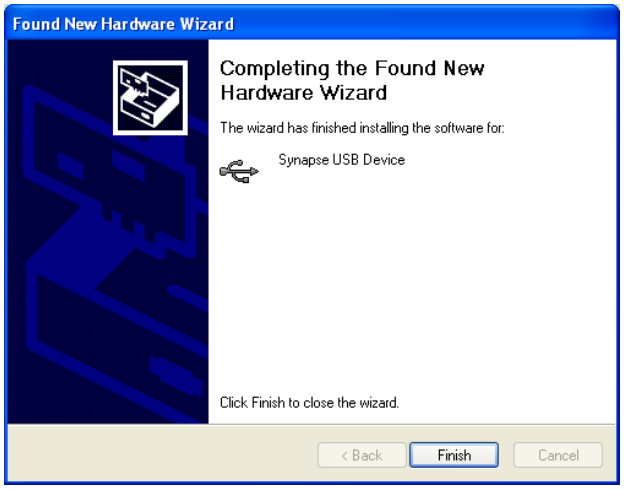


Choose "**Install the software automatically**…" and click on **Next**.

Depending on the version of Windows you are running, you may get a warning dialog similar to the following:



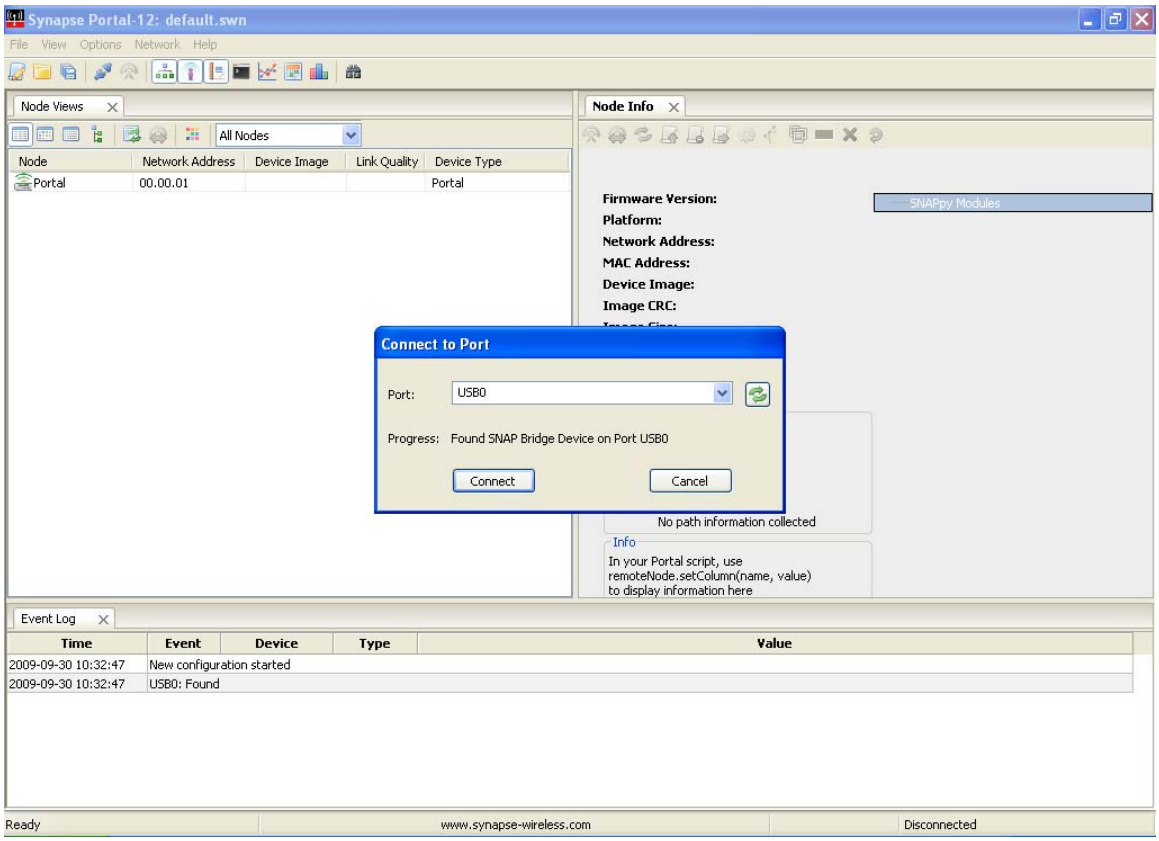
This warning is harmless, and you should click on **Continue Anyway**.



Congratulations! Your *Synapse USB Device* is installed and ready to be used by Portal.

Program Launch

After installation, launch the Portal program by double-clicking on the Portal icon on your desktop (or from the start menu). You should see a screen similar to the following:



If you do see something similar, you have now successfully installed Portal and detected a USB connected bridge device.

Note: *The Portal PC (the PC that the Portal software is running on) has no 802.15.4 radio of its own.*

One of the SNAP nodes must act as a "bridge" for it. Portal will connect directly to this bridge node, using either a USB or RS232 connection.

Portal will then be able to communicate to the rest of the SNAP nodes indirectly, by sending packets across the directly connected bridge node.

For our purposes, the SNAPstick will act as the bridge.

4. Using Portal

Starting Portal for the first time brings up a blank network configuration and a dialog asking what port to use to connect to the SNAP bridge device:

We'll continue using the SNAPstick module from the previous section, so keep it connected to the USB port.

Press **Connect** once the bridge device (SNAPstick) has been detected (this is probably going to be USB0).

Navigating within Portal

Portal is straight-forward to use. However, since it is extremely customizable by the user, your screen layout may not match the screen shots in this manual. For that reason it is important to understand the fundamental concepts used in navigating the Portal GUI.

Pull-down Menus

At the top of the Portal GUI are pull-down menus for **File**, **View**, **Options**, **Network**, and **Help** related operations.

Clicking on one of these top-level menu choices will pull up a sub-menu of additional choices. For example, clicking on **Network** will bring up a sub-menu from which you perform actions like **Broadcast Ping**, **Find Nodes…**, or **New Configuration…**. Similarly, clicking on **Help** will bring up choices such as links to the **SNAP Reference Manual**, **Synapse Support Forums**, or **License Details.**

The convention is that menu choices ending in "…" usually bring up additional menus or dialog boxes, and menu choices not ending in "…" cause immediate action to be taken, with no further prompting.

Tool Bar

Below the pull-down menus is a horizontal Tool Bar from which you can initiate several actions. Hovering the mouse cursor over each button will display a short "tool-tip" help message, and clicking on each button will actually initiate the action displayed by the tool-tip.

Tabbed Windows

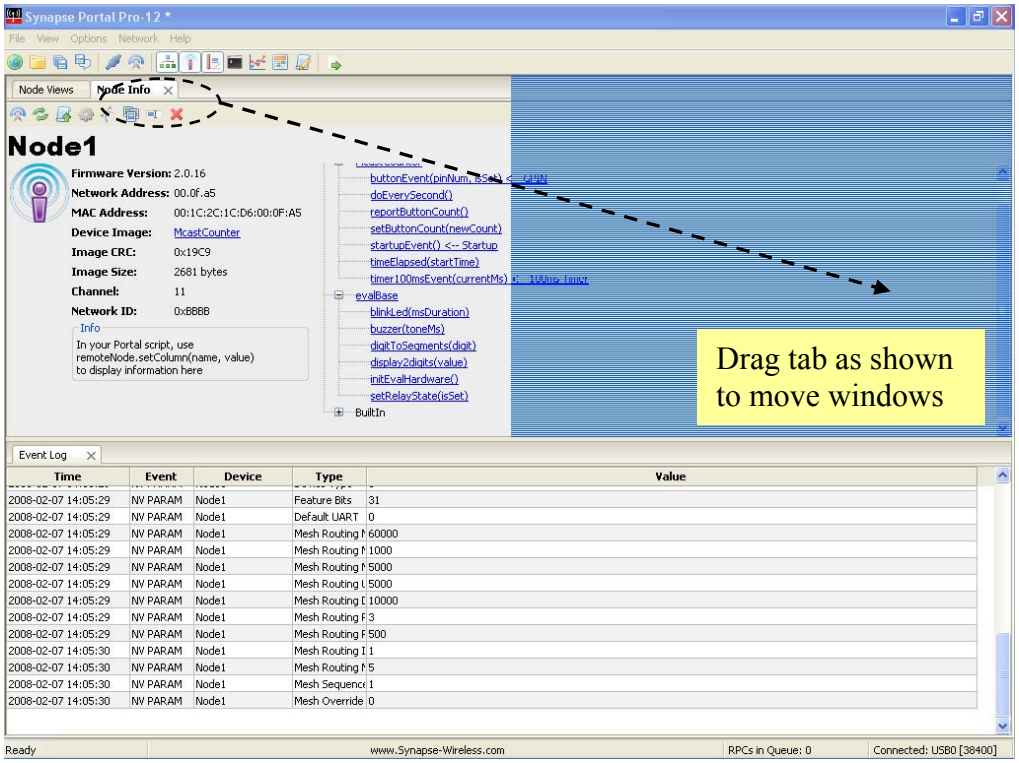
The remainder of the Portal GUI is taken up by a changeable collection of tabbed windows. Each of these windows has a name, which is displayed in the tab for that window.

Many of the tabbed windows have toolbars of their own, located in the horizontal region just below the labeled tab.

Portal starts out with an initial set of tabbed windows visible. Sometimes clicking on certain controls within one tabbed window will open and/or switch to another tabbed window. You can also open additional tabbed windows by choosing them from the **View** menu. Finally, many of the tabbed windows can be launched from the main tool bar.

Rearranging Windows

The tabbed windows in Portal can be dragged and repositioned on the screen. To do this, press and hold the left mouse button while the cursor is positioned over the tab label you want to move. While holding the button down, drag the tab until you see a light blue "shadow" indicating a possible new position for the window. When you've found a suitable new position, just release the mouse button and the move will be complete.



Resizing

Windows may be resized by clicking and dragging the horizontal and vertical borders separating them.

Closing Tabs

You can close tabbed windows that you no longer want by clicking on the small "X" located to the right of the name in the tab.

Now that you know the basics of navigating within Portal, we can continue with the tour.

Discovery

We first need to look at the **Node Views** tabbed window. If this window is not already open, you can click on **Views**, then choose **Node Views window**. Alternatively, you can click on the icon on the toolbar.

You will notice that the **Node Views** window has its own toolbar. Ignore all but the first four buttons for now.

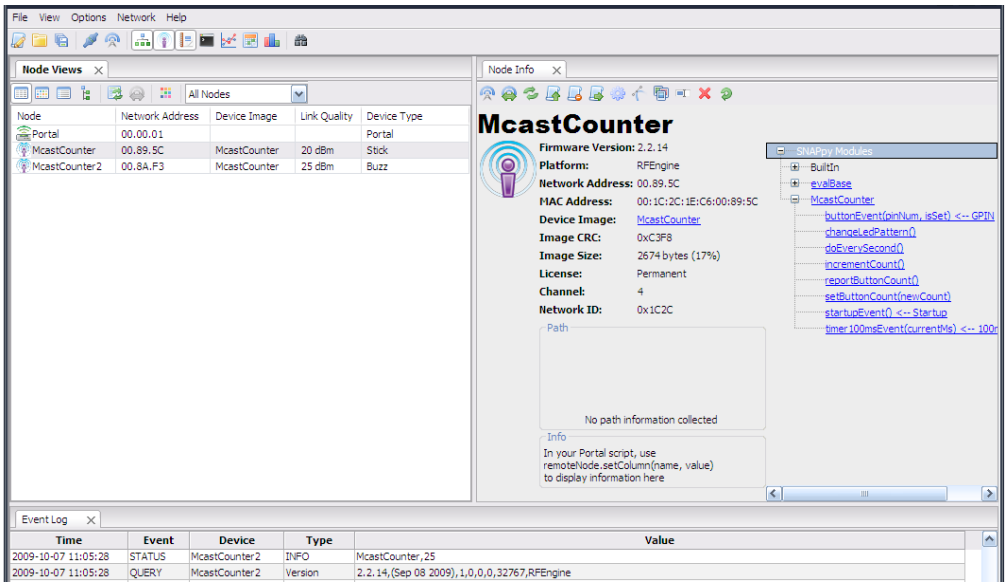
The **Node Views** tabbed window lets you look at your nodes in three different ways:

- **Report View**
- **Icon View**
- **List View**
- **Tree View**

All four views are just that, "views" of the same network information. Click on the **Report View** button.

You should see that two Devices have been discovered. They all will have names of the form McastCounterX. The directly connected "bridge" device (the SNAPstick) should be shown in blue, while the remote device should be displayed in black. Because the nodes report in using a random response delay, which node gets to be McastCounter and which node gets to be McastCounter2 can vary.

When you double-click on a node in one of the **Node Views**, Portal displays basic information about that Node in a separate **Node Info** pane.



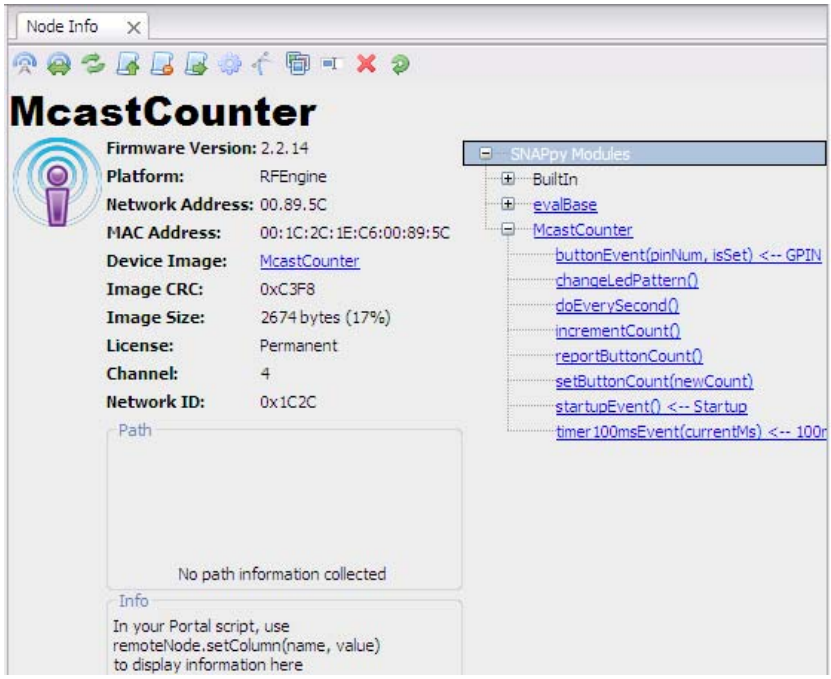
Note that the node names are based on the names of the SNAPpy scripts loaded into those same nodes *at the time of discovery*, but with additional trailing digits (2, 3, 4, etc..) added to enforce uniqueness.

Since these nodes were pre-loaded at the factory with the "McastCounter.py" script, their names are of the form McastCounterX, where "X" is replaced by a number. If the nodes had been pre-loaded with some other script, then you would have seen entirely different base names.

If the three nodes had not been pre-loaded with scripts at all, then their names would have been "Node" and "Node2".

Node Info

Lots of information is shown in the **Node Info** tabbed window.



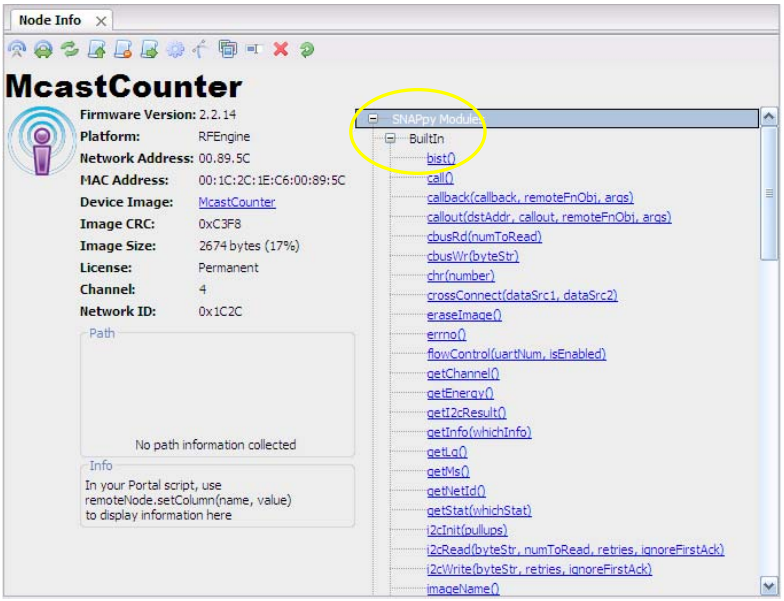
Across the top, a toolbar provides easy access to node-specific functions.

On the left-hand side, the **Firmware Version**, **Network Address**, **MAC Address**, **Device Image**, **Image CRC**, **Image Size, Channel**, and **Network Id** are shown. Additionally, there is an **Info** field which can be controlled from Portal scripts to add your own custom field(s).

Below that is a block where **Path** (the path to/from the node) information can be displayed. Below that there is an **Info** field which can be controlled from Portal scripts to add your own custom field(s) to the **Node Info** panel.

Device Image refers to the SNAPpy script loaded into the node. The term **SNAPpy Image** is used interchangeably. Here you can see that the script/image "McastCounter.py" has been loaded into the node. Also note that you can click on the device image name shown (McastCounter) and the Portal will automatically bring that script up in its built-in code editor.

On the right hand side, a collapsible tree of available functions is shown. In this next screenshot, you can see the BuiltIn tree (the tree of built-in functions) in expanded form.



Notice that there is a scroll-bar on the right-hand side of the pane – there are too many built-in functions to fit on the screen at one time.

Hovering the mouse cursor over a function name will display a tool-tip for that function. More importantly, you can click on any function to invoke that function **directly on the selected node**.

Functions that do not require any parameters (for example, the reboot() function) will be executed immediately.

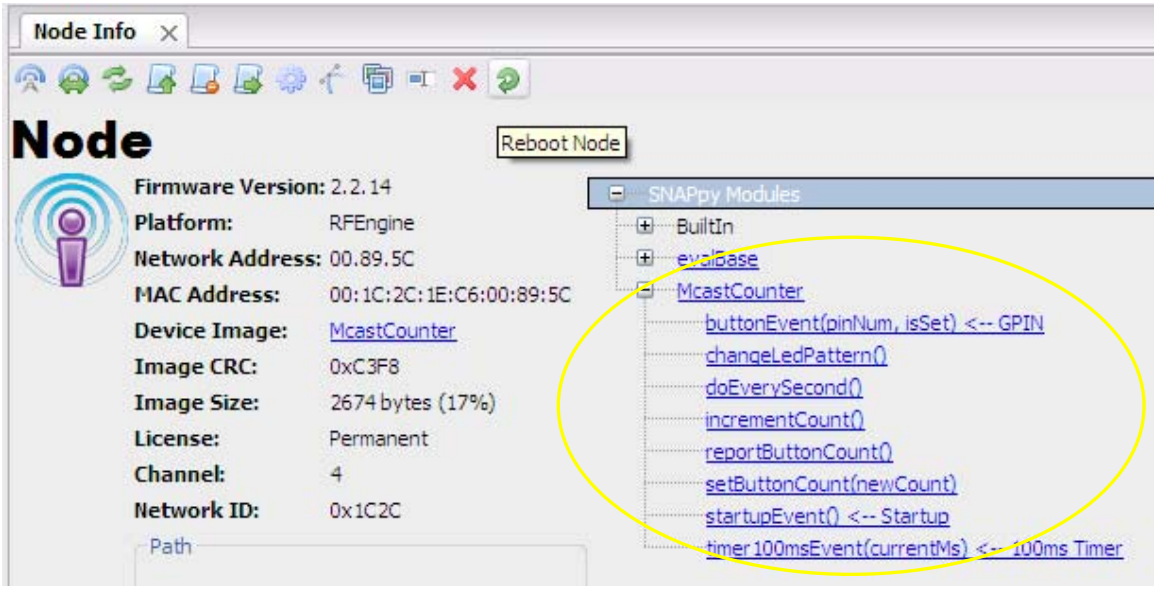
If the function requires any parameters, Portal will automatically prompt you for them.

For example, clicking on the *writePin*() function will prompt you to enter the actual value to output on that GPIO pin.

You can either:
1) Enter a value (ex. pin = 1, isHigh = True) and press OK, or
2) You can press Cancel to abort the function invocation.

Don't forget that in the Python language 'True' and 'False' are case sensitive (first letter capitalized)

You can also expand the tree of functions defined by each module (in other words, by each SNAPpy source file).



Here you can see the various functions defined in the "McastCounter.py" SNAPpy script. Like the built-in functions, these can also be directly invoked by clicking on them (and entering any needed parameters).

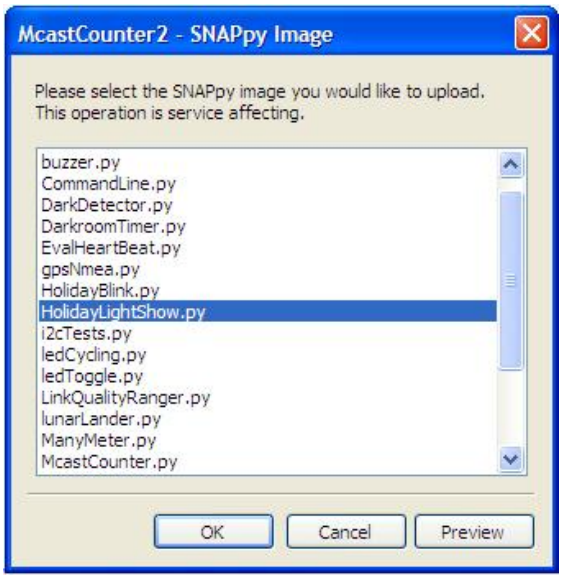
The **Node Info** tabbed window also has its own toolbar. Most of the toolbar functions will be discussed later, but one is of particular importance to us now; "Upload SNAPpy Image".

Uploading SNAPpy Images

The "multicast counter" script was preloaded merely as a convenience to the user. You can overwrite these default scripts with other scripts from the set of example scripts included with Portal, or even with your own custom scripts.

We've already tried out the "multicast counter" functionality in the first demonstration, so now let's override that behavior with some different ones. To do that, we will assign new "behavior" by giving each node a new SNAPpy Image. Select the node with Device Type "Buzz" in the *Node Views* panel (This is the type given to the proto-board). Then, in the *Node Info* panel, select "Upload SNAPpy Image" .

This will bring up a dialog box asking *which* script/image to upload.



These scripts are located in a Portal\snappyImages directory within your MyDocuments folder.

Select the "HolidayLightShow.py" example, and press OK (or you could double-click on the script name). Upload of the new SNAPpy Image (over the air!) should complete in a few seconds. The node will automatically restart when the upload finishes.

Now select the node with the device type set to "Stick" from the *Node List* and upload the "HolidayBlink.py" script into it.

> ***Note:*** *For this example we are running different scripts on each node. SNAP nodes do not have to be running the same SNAPpy script to interact with each other.*

Tutorial Time

In this section will begin to use some sensor components to demonstrate the interaction of SNAP nodes with one another, as well as, with Synapse's Portal software.

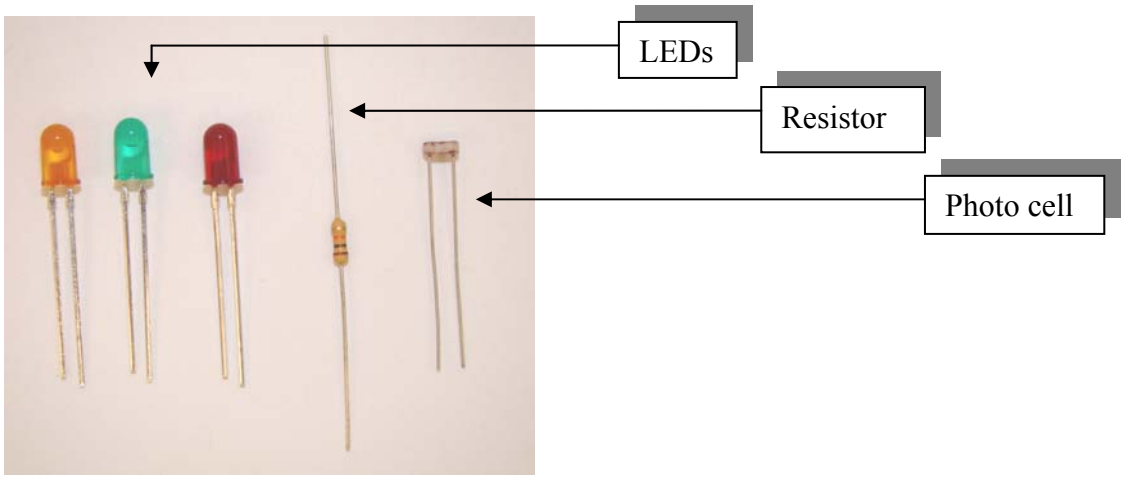
Demonstration 2: Holiday Light Show

You should now have the SNAPstick node running the example script "HolidayBlink" and the proto-board node running another example script "HolidayLightShow". If you do not, please refer to the previous section "Uploading SNAPpy Images".

Up until now we have ignored the snappy (pun intended) little Synapse screwdriver and small bag of electronic components that accompanied the kit. Now is the time to roll up our sleeves and set to work on a second demonstration.

The following components will be used:

- 1 Red, 1 Green, and 1 Amber LED
- 1 10K Ohm resistor
- 1 Photo cell



First, make sure the screws in the terminal block are set to the open position (this should be the 'factory default'). Like a standard screw, clockwise tightens and counter-clockwise will loosen the connector terminals.

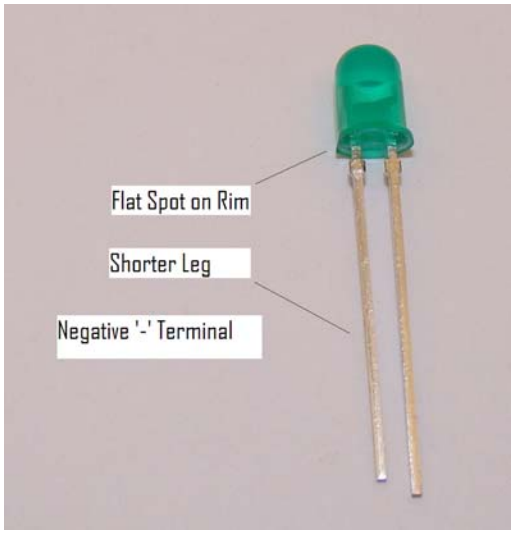
We'll use the pins on the left hand side of the proto-board to connect the LEDS. We'll also us the pins in the lower right hand corner of the proto board (GPIO11-12 and GND) to attach the photo-cell and the 10K resistor pull-up resistor for this demonstration.

> ***Note:*** *You can differentiate the included resistors by color. The 10K Ohm part is beige in color, while the 100K Ohm resistor is blue. We'll use the beige resistor for this demonstration.*

GOAL: To demonstrate how information gathered from sensors onboard a SNAP node can be communicated to other nodes and used to initiate other tasks

Step 1 – **Disconnect** the power running to the proto-board.

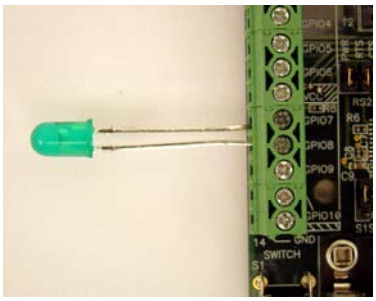
Step 2 – Determining LED polarity:
Find the green LED. LEDs do have a polarity, so we'll need to determine which end is the negative terminal (a.k.a the cathode). One of the legs of the LED should be shorter than the other. This is the "negative" leg. A flat spot on the rim at the base of the colored bulb also indicates which side is the negative. The following picture should help:



Step 3 – Connecting the green LED:
Place the "negative" leg of the green LED into the terminal block at the pin labeled GPIO 8 (the fourth pin up from the bottom of the left-hand side of the proto-board).

Place the other leg of the green LED into the terminal block at the pin labeled GPIO 7

Tighten the screws located on the top of the terminal block for both GPIO pins.



Step 4 – Connecting the red LED:
Place the "negative" leg of the red LED into the terminal block at the pin labeled GPIO 6 (near the middle of the left-hand side of the proto-board).

Place the other leg of the red LED into the terminal block at the pin labeled GPIO 4. Note that since this is not the adjacent slot on the connector you might have to bend (slightly) the legs of the LED (See picture).

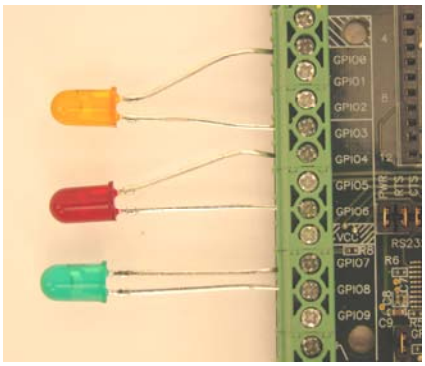
Tighten the screws located on the top of the terminal block for both GPIO pins.

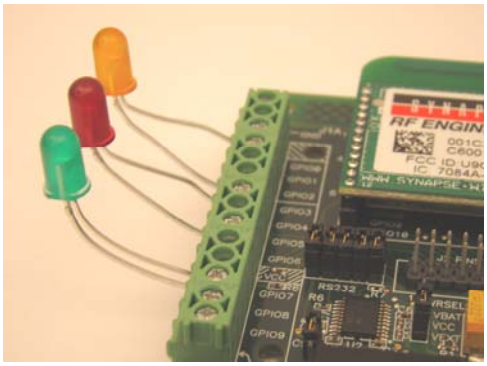
Step5 – Connecting the amber LED:
Place the "negative" leg of the amber LED into the terminal block at the pin labeled GPIO 3.

Place the other leg of the amber LED into the terminal block at the pin labeled GPIO 0. Note that, like before, we are dealing with non-adjacent slots on the connector. You might have to slightly bend the legs of the LED.

Tighten the screws located on the top of the terminal block for both GPIO pins.

Aside: You can bend the LEDs up to better see the light show.





Step6 – We are now ready to interact with the Portal software we installed in the previous section of this document (if you have yet to install Portal, please go back now).

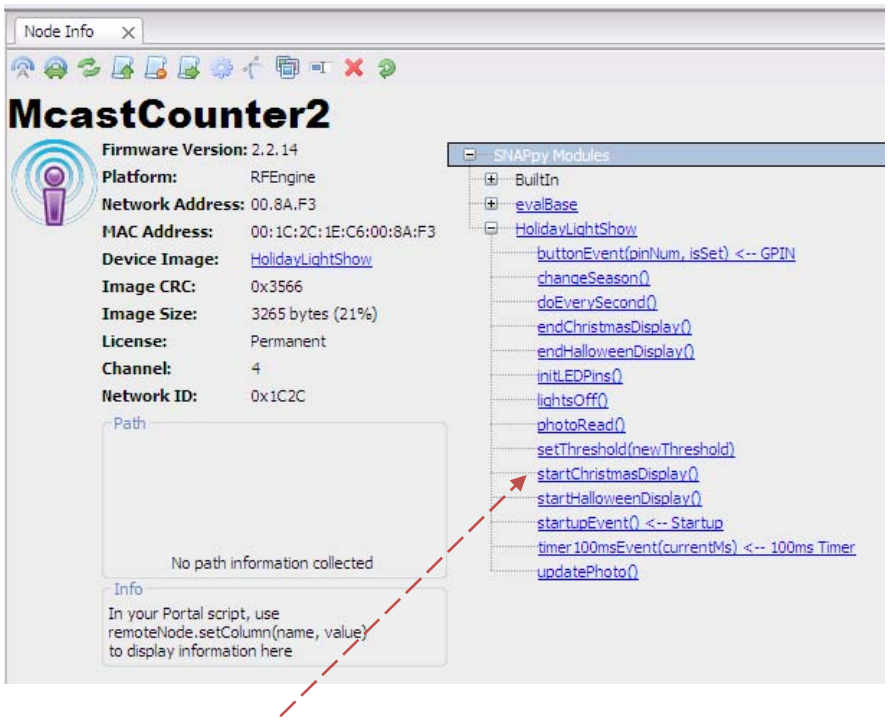
Time to check on the quality of our work by testing the LEDs. We can use the new SNAPpy script we just uploaded to the node for this purpose.

Connect the power back to the proto-board.

Switch to the *Node View* pane in Portal that was described in the previous section.

Select the device with the *Device Type* set to "Buzz" (it will be running the "HolidayLightShow" script).

Switch to the *Node Info* pane and find the "HolidayLightShow" specific section from list of available functions. You might need to collapse the "evalBase" section of functions.



Click on the *startChristmasDisplay*() function.

You should see the green and red LEDs begin to blink.

Click on the *changeSeason*() function.

You should see the amber and red LEDs begin to blink. (The green LED is now off).

Click on the *lightsOff*() function.

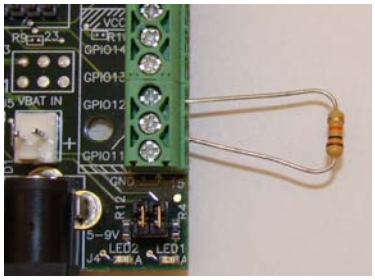
All 3 LEDs should now be off (we are ignoring the proto-board's built-in yellow LED).

If you did not see each of the LEDs flash at some point during the test, check you connections. Try a gentle tug on each LED. They should not move, but be firmly connected to the device.

For the time being we'll need to **disconnect** the power to the proto-board

Let's take a look at what we just did. The Portal application running on your PC was able to wirelessly communicate with the remote device through the bridge node. We were even able to execute functions specific to the script running on the remote device. How cool is that?

Step7 – Connecting the resistor:
Bend the legs of the beige resistor towards one another as seen in the picture

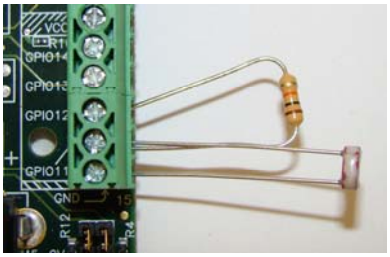


Place one resistor leg into the terminal block at the pin labeled GPIO 12 (the third pin up from the bottom of the right-hand side of the proto-board).

Place the other leg into the connector at GPIO 11. There is no polarity on this resistor, so it does not matter which leg you choose for which pin.

Tighten the screw for GPIO pin 12 only. This should lock the one leg of the resistor in place. We'll address the other leg in a moment.

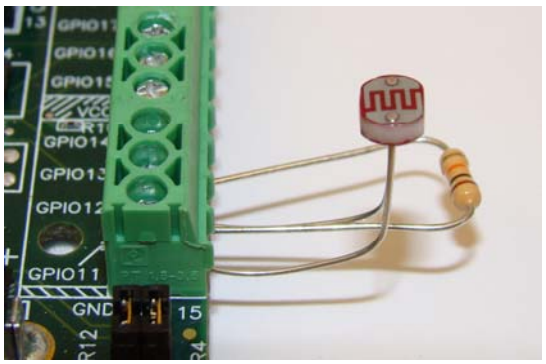
Step 8 – Connecting the photo-cell:
Place one leg of the photo-cell into the pin labeled GPIO 11 (don't worry it is supposed to share the connector with the resistor) and the other leg in the last slot in the connector (the GND pin). There is no polarity on the Photo cell, so it does not matter which leg you choose for which pin.



Tighten the associated screws and try a gentle tug on the components. They should not move, but be firmly connected to the device.

Carefully **bend** the photo cell so that the face of the component points up. Refer to the pictures; they really can be worth a thousand words.

Aside: Feel free to trim the leads of any of the devices to provide a 'tidier" setup.



***Note**: While not used in any demo scripts, there are software controlled internal 25 K Ohm pull-up resistors for each input pin. These can be controlled using the setPinPullup() function. See the SNAP Reference Manual for more details.*

Step 9 – **Reconnect** the power running to the proto-board.

Step 10 - The "HolidayLightShow" script is *already* monitoring for changes in light levels on the proto-board (sneaky, I know). However, the script includes an "auto-calibration" capability and since we have not "calibrated" the sensor yet, no values will be acted upon.

Place your finger over the photo-cell to represent "complete darkness". This will give the script a max reading for calibration. Now, if you pull your hand away, the sensor will be reading the relative darkness.

Step 11 **– Let's start a Christmas light show.**
Hold your hand close to the sensor. The red and green LEDs will begin to flash once your hand casts a dark enough shadow over the photo cell.

Look over at the SNAPstick. You will notice that the green and red LEDs are lit up as well. The proto-board has communicated the change in sensor status over the air to the neighboring device to start the Christmas show.

Aside: Sometimes holding your finger close to the sensor will trigger it twice. If you see the orange and red LEDs instead, you can move your hand away and then close again to transition back to the green and red lights.

You can change the trigger point (light-intensity value) at which the script begins/changes the light show using the *setThreshold*() function in Portal (default is 85).

Step 12 **–** Time to change the season.
Move your hand away and then back to hovering over the sensor again. A Halloween light show (orange and red LEDs) will begin to flash. As before, the proto-board has wirelessly communicated the message to change the light pattern to the SNAPstick.

Oh, one final thing: the button on the bottom of the proto-board will turn off the lights or you can execute the *lightsOff*() function on the proto-board using Portal.

Success! We now have a device that will sense when the sun goes down and begin a Christmas light show one night and a Halloween light show the next.

KEY POINTS:

- Each node can be controlled by Portal using a bridge node (although the nodes can function without the presence of Portal)
- Portal can be used to execute script functions on remote nodes as though it were local to the node.
- SNAP nodes do not have to be running the same SNAPpy script to interact with each other.

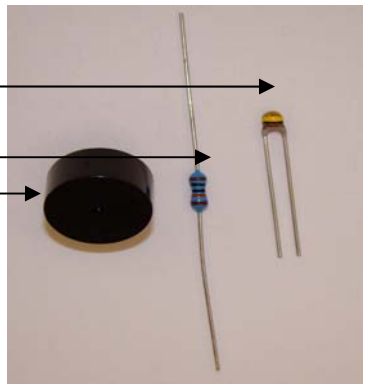
> ***Taking a deeper look:*** *If you peek at the HolidayLightShow SNAPpy script you will see how easy it is for SNAP nodes to communicate. It contains a line of code: "mcastRpc(1,2,"christmasBlink")". This single line is all it takes for a SNAP node to send a multicast message to a group of devices.*
>
> *This particular example uses a multicast Remote Procedure Call (RPC). A uni-cast RPC can be sent to a specific unit using SNAP's built-in rpc() function. More information about SNAPpy scripting can be found in the SNAP Reference Manual.*

Demonstration 3: Temperature Alarm

The proto-board still looks a little dull. Let's attach some more "fun stuff". This time we'll connect a standard thermistor and a 100KΩ pull-up resistor.

The following components will be used:

- 1 Thermistor
- 1 100K Ohm Resistor
- 1 Buzzer



GOAL: To further demonstrate the interaction of individual nodes with each other and with Portal.

Remember, just like a standard screw, clockwise tightens and counter-clockwise will loosen the connector terminals.

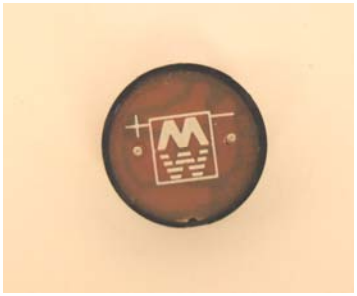
We'll use the pins in the upper right hand corner of the proto board (GPIO 18, VCC, and GND) to attach the thermistor and the 100K pull-up resistor for this demonstration.

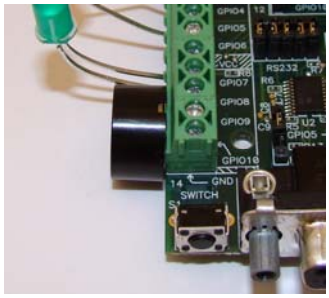
> ***Note:*** *You can differentiate the included resistors by color. The 10K Ohm part is beige in color, while the 100K Ohm resistor is blue. We'll use the blue resistor for this demonstration*.

Step 1 – **Disconnect** the power running to the proto-board.

Step 2 – Connecting the buzzer to the proto-board:
Place the leg of the buzzer (see picture) marked with a '+' into the connector at GPIO pin 9 and the other leg into the connector marked GND.

Tighten the associated screws and try a gentle tug on each of the components. They should not move, but be firmly connected to the device.





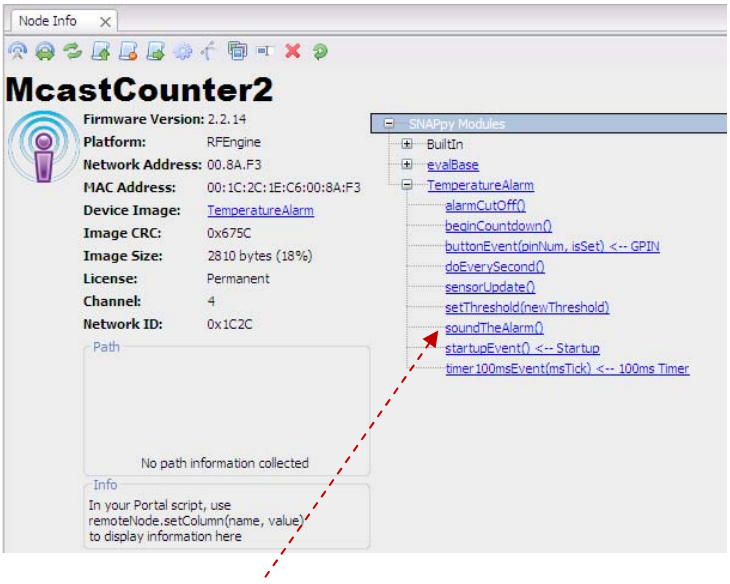
Step 3 – We'll need to upload another set of scripts to the nodes using Portal:
Connect the power back to the proto-board.

Load the script "TemperatureAlarmBridge.py" onto the SNAPstick (i.e. the unit with the *Device Name* set to "Stick") using the step you learned in the "Uploading SNAPpy scripts" section.

Load the script "TemperatureAlarm.py" onto the proto-board (i.e. the unit with the *Device Name* set to "Buzz")

Step 4 –
Make sure that you still have the proto-board selected in the *Node View* pane.

Switch to the *Node Info* pane and find the "TemperatureAlarm" specific section from list of available functions.



Click on the *soundAlarm()* function to test the buzzer. You should hear a long beep from the buzzer.

Disconnect the power running to the proto-board again.

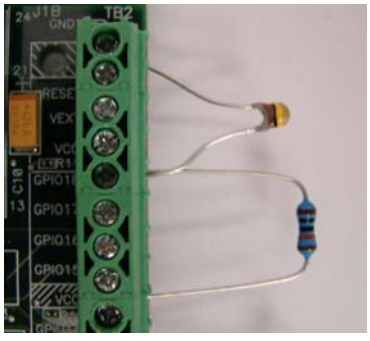
Step 5 – Connecting the resistor (blue) to the proto-board:
Bend the legs of the resistor towards one another as seen in the picture

Place one resistor leg into the terminal block at the pin labeled GPIO 18 (located on the right-hand side of the proto-board).

Place the other leg into the connector at the pin labeled VCC (located in the center of the right-hand side). There is no polarity on this resistor, so it does not matter which leg you choose for which pin.

Tighten the screw for the VCC pin only. This should lock the one leg of the resistor in place. We'll address the other leg in a moment. (Déjà vu?)

Step6 – Connecting the thermistor to the proto-board:
Place one leg of the thermistor (see picture for identification) into the pin labeled 18 (don't worry it is supposed to share the connector with the resistor) and the other leg in the first slot in the connector (the GND pin). There is no polarity on the thermistor, so it does not matter which leg you choose for which pin. Again, a picture is worth a thousand words, so please refer to the figure for how the setup should look. Notice how you have to bend the legs of the thermistor to reach across several connector slots.



Tighten the associated screws.

Step 7 – **Connect** the power back to the proto-board.

Step 8 – **Hold** the thermistor between your thumb and forefinger to raise the temperature. Once it reaches a pre-set threshold it will trigger a 5 second timer. The start of the timer will be indicated by the green LED on the proto-board and a flashing amber LED on the SNAPstick.

The buzzer will sound once the timer expires unless the alarm cut-off is executed. This can done one of two ways:

1. Executing the *alarmCutOff*() function in the "TemperatureAlarm" specific section of the *Node Info* pane within Portal (this can be done on either node).
2. By pressing the built-in proto-board button.

A quick 'chirp' of the buzzer will indicate that the alarm has been bypassed (much like your car alarm).

KEY POINTS:

- Synapse RF engines have the ability to read analog inputs.
  - Each engine has 8 analog inputs included in the 19 GPIO pins.
- Remote nodes can use sensor or other inputs to trigger actions on other remote nodes. The alarm was triggered on both nodes and could be disabled by either node.

5. Portal's Extended Capabilities

Built-in Functionality

One of the key elements of a sensor network is being able to log and/or display the gathered information.

Portal has several built-in ways to track and/or display data. Let's use the sensors we just installed on the proto board in another example.

Demonstration 4: The Many-Meter

Well, you have probably heard of the multi-meter. Let's put together a many-meter. The good news is that we already have all the components we will need wired up and ready to rock.

GOAL: To show how information can be communicated back to the Portal PC for display and logging.

Step 1 – **Switch** to the *Node View* pane in Portal and **select** the node associated with the proto-board.

Upload the "Many-Meter.py" SNAPpy script onto the proto-board (a skill we've mastered from previous sections, but here is a hint: .).

Let's give the proto-board a name. **Click** on the *Change Configuration* icon from the toolbar and then select the *Device* tab. **Change** the *Device Name* (currently blank) to read "OurDemo". **Click** "OK".

We must **restart** the node for the name change to take effect. You can do this by **clicking** on the *Reboot Node* icon from the same toolbar.

Step 2 – **Switch** back to the *Node View* pane and **select** the node associated with the SNAPstick.

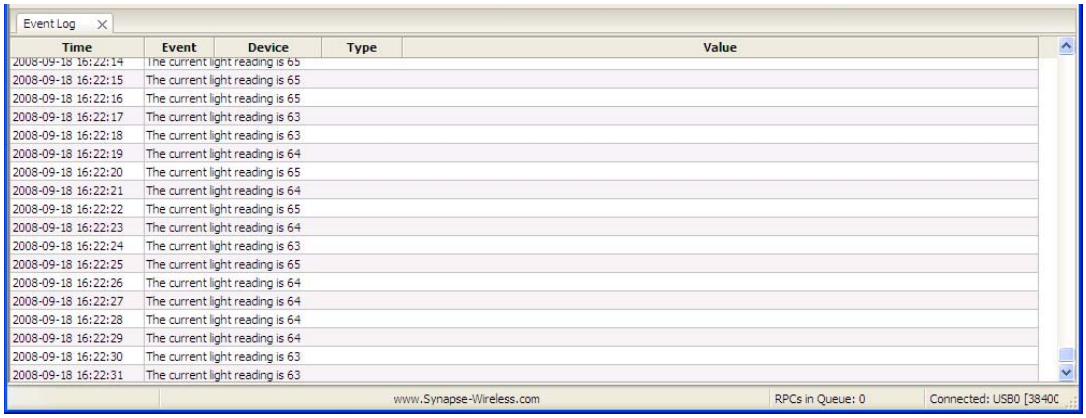
Switch to the *Node Info* pane and **Erase** the script currently running on the SNAPstick by clicking on the icon located in the toolbar.

Select the *New Configuraton…* option from the Network pull-down menu in Portal. This will reset your view to use our new name. The SNAPstick without a script will now have a node name like "node1" or "node2".

> ***Note:*** *You do not need to have a SNAPpy script running on the SNAP node for it to participate in a SNAP network.*
>
> *In fact, Portal can still configure and interact with remote nodes regardless of the existence or type of script running on the bridge node or remote nodes.*

Step 3 – **Open** the event-log pane. You'll see the remote node is forwarding the current light-intensity reading from the photo-cell to be logged by Portal. This is actually a "darkness" reading since it will increase as it gets darker.

Calibrate the sensor once again by covering it to simulate "complete" darkness.



Step 4 – **Push** the button located on the proto-board and watch the data change from light-intensity to the temperature reading. (This is a raw measurement where the value from the thermistor actually decreases as the temperature increases)

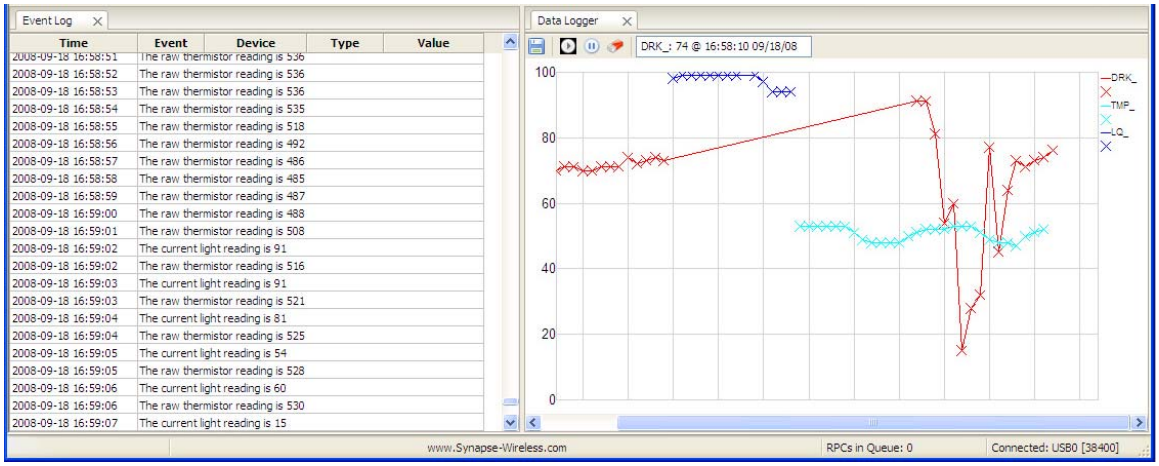
Push it again and the radio link quality will be displayed.

Push it one more time (is your thumb sore yet?) and you will see the "darkness" reading and thermistor reading displayed at the same time.

Step 5 – **Open** the *Data Logger* pane in Portal. The same information that was displayed in the event log should also be given in a graphed format. This information can be saved by clicking on the 'save' icon in the upper right hand corner of the pane. The graphed data can be paused, restarted, or cleared from the same toolbar .

> ***Remember*** *you can move the position of the Data Logger pane (or any other pane) within Portal by using your mouse to click and drag. This was discussed in the Using Portal section.*

Press the button on the proto-board to cycle through each of the available sensor readings and watch Portal graph the associated information. Notice that multiple readings can be displayed concurrently. Every 4th button press will disable the display of information.



KEY POINTS:

- SNAP nodes do not require a script to participate in a SNAP meshed network.
- Portal has a built-in *Event Log* and *Data Logger* to display information gathered by any number of SNAP nodes.

Taking a deeper look:
The ability to present event-log messages and display information in the data-logger can be accomplished using calls to functionality built into the Portal node.

If you peek at the file "ManyMeter.py" you will see this SNAPpy script forwards data to Portal and instructs it to post it to the log by using a Remote Procedure Call (RPC). The script will contain a single line of code that reads: "rpc(portalAddr, "logEvent", eventString)". This example uses a uni-cast RPC (single address destination) to call a function directly within the Portal node. This is explained in detail within the SNAP Reference Manual.

Advanced Functionality:

Ok. We've reached a point where we want to step it up a notch.

The next section describes functionality that is available to the user, but is not necessarily a part of the Portal software application.

WARNING: What lies ahead is something we feel is extremely powerful and, as such, something we should at least touch on. However, it is not for the faint of heart. It involves a little bit of computer science and programming knowledge. So, proceed with caution….

Portal and the SNAPpy scripts that run on each node are based on a sub-set of the Python programming language. (More on Python can be found at www.python.org). A Python interpreter is already running in order to support Portal. This includes a python open-source library called wxPython which provides Graphical User Interface (GUI) support (Info at: wxpython.org).

The fact Portal is already running this library allows us to dip into the same functionality. We can create our own custom graphical output through Python scripts running on the Portal Node.

What you are about to see is *NOT* a part of Portal, but is a library extension available through Portal. In fact, you can install and run Python with the wxPython libraries on any PC. Portal does not need to be installed to use them.

Demonstration 4b: The Many-Meter extended

That being said, let's continue with the Many-Meter example we should already have up and running.

> ***Note:*** *In the previous demonstrations we have only uploaded scripts to SNAP devices (ie. the devices containing RF engines). However, the Portal software can also act like a node in the SNAP network by using the connected bridge node to send and receive network traffic.*

Step 1 – **Switch** back to the *Node View* pane in Portal and **Upload** the script call "PortalManyMeter.py" to the Portal node. This file will be located in the "My Documents\Portal" folder.

Step 2 **–** From the same *Node View*, **verify** that the proto-board (node with the *Device Name* set to "Buzz") is still running the "ManyMeter" script. **Click** on the node.

Switch to the *Node Info* pane and click the script name next to the *Device Image* heading. This will bring up the SNAPpy script for editing within the Portal environment. An alternative is to use the *Open File…* command from the *File* drop-down menu.

Step 3 – Edit the SNAPpy script:
The original script is read-only. So, **Click** on the '*Save As*' icon  and save a copy of the file with the new name "ManyMeterPlus.py".

Find the line of code with the comment "EDIT NEXT LINE". The very next line is in fact a call to the *rpc()* function and has been commented out (Python comments start with the '#' character). We've cause to use it now, so let's un-comment it.

Original Code:
```
# EDIT NEXT LINE: This is special code to call into the wxPython functionality of Portal
#rpc(portalAddr,"DisplayData",photoVal,"Dark Meter",loadNvParam(NV_DEVICE_NAME_ID))
```

Delete the single '#' character from the beginning of the line.

Modified Code:
```
# EDIT NEXT LINE: This is special code to call into the wxPython functionality of Portal
 rpc(portalAddr,"DisplayData",photoVal,"Dark Meter",loadNvParam(NV_DEVICE_NAME_ID))
```

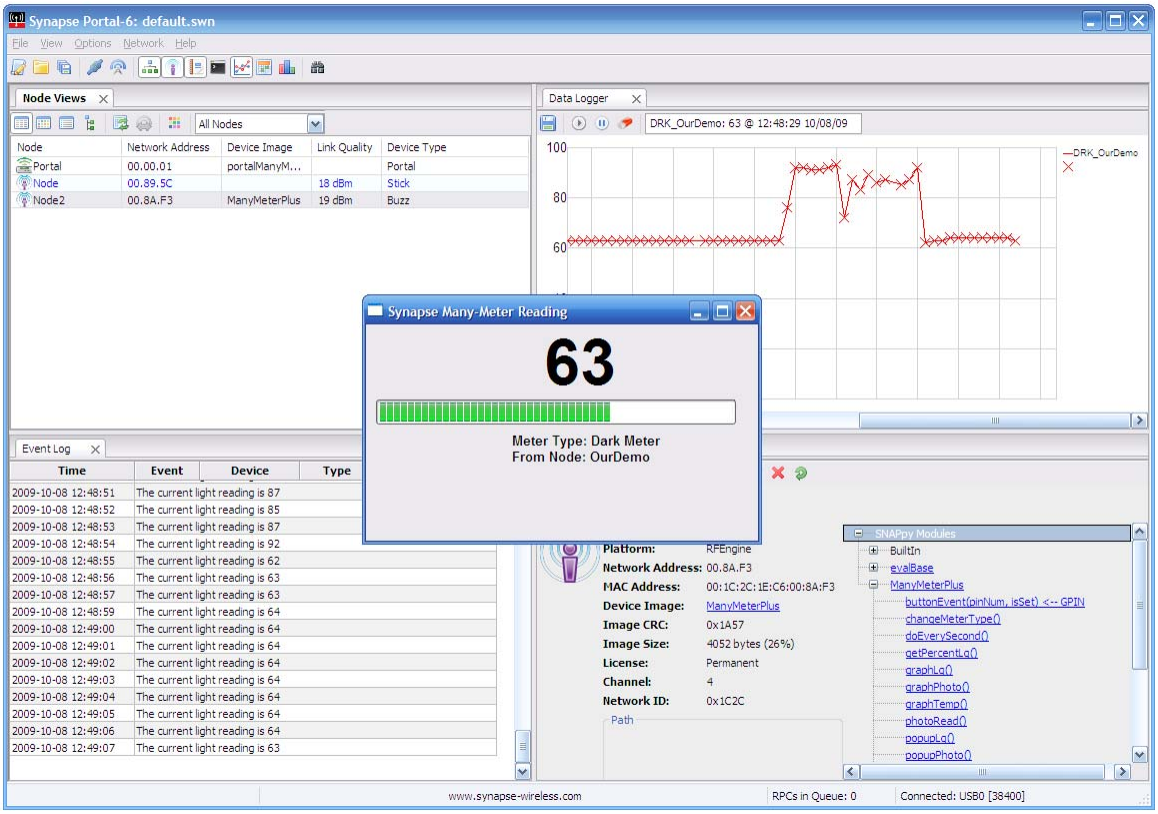
Repeat this process for the other 2 lines (for a total of 3 lines) with the same comment.

Save your work by clicking on the 'Save' icon  in the toolbar.

Step 4 – **Upload** your brand new script, "ManyMeterPlus.py", to the proto-board node (device name set to "Buzz").

You should now see a separate window that displays the current sensor readings in a graphical form.

To get true 'darkness' readings you will need to calibrate the light sensor once again by holding your finger over the sensor head.



Look at the code snippet we just edited: you will see that the remote node is performing a RPC call into the function "DisplayData" on the Portal node. It is the "DisplayData" function that accesses wxPython functionality on the PC.

KEY POINTS:

- The Portal node can run scripts like other SNAP nodes.
- SNAP scripts can use the Portal node to access extended functionality. This can include such things as:
    - Email
    - Output to database or log files
    - Graphical output to the screen

***Taking a deeper look**:*
The Event Log message settings can be configured from the options menu (options->configure logging). You'll notice from this menu that Portal can be configured to send information via email. This functionality is also available to scripts running on the Portal node. In other words, you can configure scripts to email different addresses based on observed events and conditions. More details are available within the SNAP Reference Manual.

6. Alternative Energy Settings

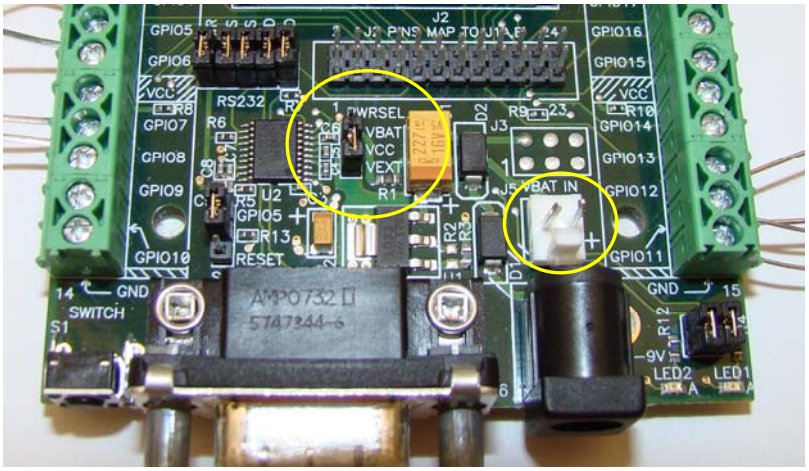
Battery Operation

So far, we have been running the nodes from USB power (the SNAPstick) and from wall power (the SN171 proto-board).

It is also possible to run the SN171 proto-board from battery power using the battery holder included in the EK2100 kit.

Before you can power this board from the external battery pack, you must unplug the external power supply, and change a jumper located near the center of the circuit board.

The jumper posts for this jumper are labeled “VBAT”, “VCC”, and “VEXT”.



The board comes from the factory with the jumper connected to the VEXT and VCC pins, which configures the node to run from the external power supply.

If you move the jumper so that it connects the VBAT and VCC pins, then the board will be configured to receive power from the white two-pin connector located directly behind the barrel-jack that the external power supply plugged into.

After changing the jumper to the VBAT+VCC position, install two AA batteries in the external battery holder, and connect the white connector from the external battery pack to the mating white connector on the proto-board board.

NOTE! There is an on/off switch on the external battery pack. Be sure to slide it to the “On” position when you want to power up the SN171 proto-board.

Low Power Operation

The SN171 Proto-Board is capable of achieving *years* of battery life from the included AA pack, but to do so requires running a SNAPpy script which *sleeps*, as well as removing **all** RS232 jumpers (JMP2, JMP5, JMP6, JMP7 and JMP8).

Unless you are running such a low-power script, be sure to turn battery power **off** when the node is not in use.

See example SNAPpy script "protoSleepCaster.py" for one example of low-power operation. This script is like McastCounter.py, but sleeps between button presses.

7. Where To Go Next

In this manual we have introduced the components of the EK2100 Evaluation Kit, installed Portal, and run some simple demos.

Now you will want to take advantage of some of the other SNAP documentation:

- The "SNAP Reference Manual"
- The "SNAP Hardware Technical Manual"
- The "SN171 Proto Board Quick Start Guide"
- The "SNAPstick USB Module Quick Start Guide"

These documents are in Portable Document Format (PDF) files on the CD included with the evaluation kit.

Are you still craving more?

Details regarding other kits and hardware can be found online at http://www.synapse-wireless.com under the *Products* heading.

Synapse also offers a larger evaluation kit called the EK2500. It includes two other hardware demonstration boards and different application demos. The expanded capabilities of this kit allow you to further explore how SNAP nodes interact with one another as well as Portal.

The EK2500 kit includes two demonstration boards not included in this evaluation kit:

- The SN163 Bridge Demonstration Board
- The SN111 End Device Demonstration Board

You can also purchase any of these boards individually (outside of kit form) as additional nodes. They will interact with any of the included EK2100 demonstration boards.

The EK2100 kit only includes a single type of Synapse RF engine. Other forms of the RFET and RFE that were mentioned in the introduction of this document can also be purchased on an individual basis.

Remember: Any of the RF engines can be used in interchangeably with any of the demonstration boards.

Be aware that the Portal software included with the EK2100 kit is limited to 6 nodes. More licenses can be purchased if you are interacting with a larger network

Visit us online

For more information about Synapse, SNAP networking, and SNAP product offerings, please visit:

http://www.synapse-wireless.com

You can find an ever-expanding collection of useful information on the Synapse Support Forum at http://forums.synapse-wireless.com, including:

- Quick start guides for all Synapse hardware
- Synapse application notes
- More example scripts
- Software Updates
- Question and answer discussions

The form allows you to see questions and answers posted by other users, as well as giving you the ability to post your own questions.