USERS GUIDE

DK-200

200 Series Core IoT Development Kit

For SNAP and Portal Version 2.5

Document Revision v1.0

Doc# 116-031520-001-A000

6723 Odyssey Drive // Huntsville, AL 35806 // (877) 982-7888 // Synapse-Wireless.com

Disclaimers

Information contained in this Manual is provided in connection with Synapse products and services and is intended solely to assist its customers. Synapse reserves the right to make changes at any time and without notice. Synapse assumes no liability whatsoever for the contents of this Manual or the redistribution as permitted by the foregoing Limited License. The terms and conditions governing the sale or use of Synapse products is expressly contained in the Synapse's Terms and Condition for the sale of those respective products.

Synapse retains the right to make changes to any product specification at any time without notice or liability to prior users, contributors, or recipients of redistributed versions of this Manual. Erraholidta should be checked on any product referenced.

Synapse and the Synapse logo are registered trademarks of Synapse. All other trademarks are the property of their owners. For further information on any Synapse product or service, contact us at:

Synapse Wireless, Inc.
6723 Odyssey Drive
Huntsville, Alabama 35806
256-852-7888
877-982-7888
256-924-7398 (fax)

www.synapse-wireless.com

License governing any code samples presented in this Manual

Table of Contents

1. Before Getting Started

You've come to the right place: This manual is a tutorial introduction to the Synapse SNAP product line, and you should definitely read through it and try out all of the hands-on examples it contains.

When you have completed this manual, you will have:

- gotten familiar with the included SNAP hardware
- installed the companion Portal software and any needed device drivers
- gotten familiar with the basics of using Portal and SNAP to develop wireless applications.

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.

This manual 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.

Finally, be aware that this manual is a starting point if you will, just one piece of a much larger set of documentation.

Other Documentation

This document, the DK-200 Users Guide, is only one of several featured with this evaluation kit. (Several documents are also installed on your system when you install Portal, which you will do as part of this tutorial.) Be sure to also take a look at:

- The "SNAP Primer"
  This document contains an introduction to SNAP and explanations of how mesh networking works. It also introduces the various Synapse software and hardware available, and clarifies the naming conventions used for the various SNAP items.

- The "SNAP Users Guide"
  This document is where you will find an explanation of how the components in a SNAP network work together, with introductions to topics like hook handling and with sample scripts.

- The "SNAP Reference Manual"
  This document is where you will find information on the built-in functions provided by Portal and SNAPpy. It also provides information specific to each platform to which SNAP has been ported.

- The "SNAP Sniffer Users Guide"
  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.

- The "Portal Reference Manual"
  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.

Additionally, you may wish to refer to the following hardware guides:

- The “SNAP Hardware Technical Manual”
- The “SN171 Quick Start Guide”
- The “SN172 Quick Start Guide”

All of these documents are in Portable Document Format (PDF) files and are available on the Synapse Wireless forum website.

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 has examples and Application Notes, waiting to be downloaded.

The forum also contains all the latest copies of the documentation included with this kit, plus the latest versions of Synapse Wireless software. Be sure to download the newest version of Portal (which includes the most recent firmware) for the latest feature set.

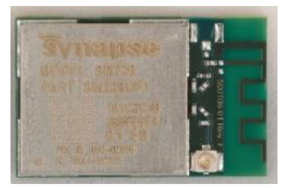
Also be sure to check out the Synapse website at www.synapse-wireless.com.

2. Getting Started

Overview

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

SNAP is the name of our network operating system. The word 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 included in this kit is a Synapse SNAP Engine.

Each SNAP Engine combines a microcontroller, a radio, and an antenna. The antenna can either be an integral chip antenna, or a mounting point for an external antenna. The SNAP Engines included in your DK-200 are RF200, RF266 and SM220 SNAP Engines, as pictured above. These SNAP Engines communicate using 802.15.4 2.4 GHz radio signals.

Each SNAP Engine has an on-board microcontroller with its own internal RAM and ROM. No external components are required for operation. SNAP Engines also include General Purpose I/O (GPIO) pins, which can be configured as digital inputs or outputs. Many of these same GPIO pins can also be switched to alternate functionality.

The exact number pins and how they can be repurposed will vary from platform to platform, but each SNAP Engine will be able to support:

- analog inputs, providing 10-bit resolution (or better)
- serial data lines (one or two UARTs, depending on platform)
- serial handshake lines (one RTS and one CTS per UART).

Although all the 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 SNAP Engines contain core code (written in C) that implements basic wireless networking functionality. This core code also implements a virtual machine that executes a subset of the Python programming language. Synapse has named this subset of Python SNAPpy.

Note: You can find details on the SNAPpy language, and how it compares to Python, in the "SNAP Users Guide." For now, the important point is to understand that SNAP nodes support a scripting language.

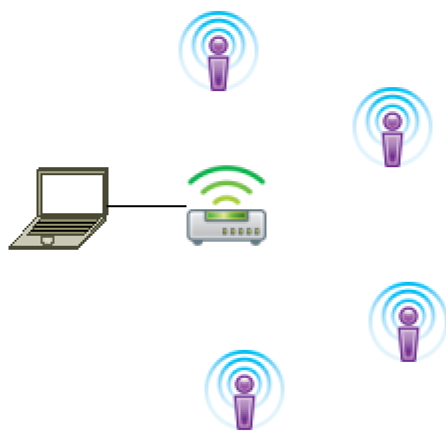
SNAPpy scripts can be uploaded into SNAP Engines “over the air” (OTA), or over the serial interfaces. These scripts define the personality of each node; by changing the SNAPpy script in a node you change the node’s behavior.

SNAP Engines can be designed into your own products, acting as a slave device to your main microcontroller or microprocessor. In many cases the SNAP Engine can take over the functionality of the original main processor, in addition to adding wireless capability and SNAPpy scripting. In addition to discrete SNAP Engine modules, Synapse also sells SNAP demonstration boards that extend the basic core capabilities with additional I/O hardware. Three different kinds of SNAP demonstration boards are included in this evaluation kit:

- One RF266 USB Dongle
- One SN171 ProtoBoard
- One SN172 ProtoBoard.

A Portal into Your Network

Portal is a standalone software application that runs on any standard PC with Microsoft Windows 2000 or higher. Using a USB or RS232 interface, it connects to any SNAP 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 (by default) within Portal to generically represent a SNAP node.

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

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.

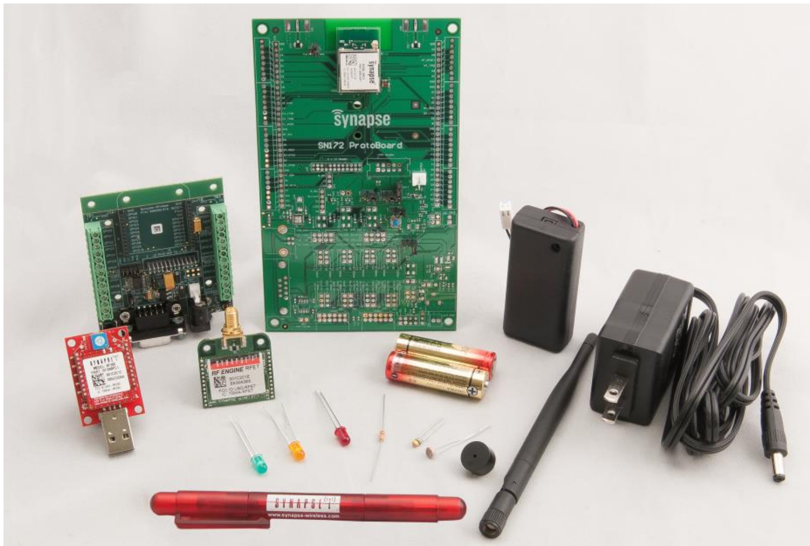
Intelligent – the network is purpose-built for your application, with the ability to monitor and conditionally control things connected to it.

Synapse’s Portal administrative software must be downloaded from Synapse’s website http://forums.synapse-wireless.com. For more information about using Portal, refer to the Portal Reference Manual available from our website. Throughout this manual, the term “Portal PC” is used to refer to the PC that Portal is running on.

Note: Synapse also licenses a standalone SNAP Connect library, giving you access into your SNAP network. Import the SNAP Connect library into your Python application to allow your backend systems to participate seamlessly in the SNAP network.

Development Kit Hardware

To demonstrate many of the capabilities of Synapse SNAP Engines, SNAP, and Portal, we've bundled them together in an evaluation kit form – the DK-200.



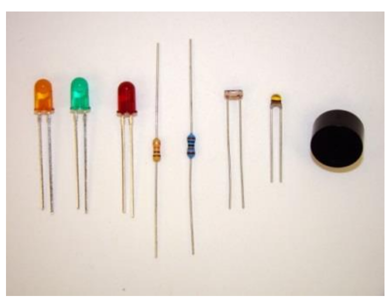
The DK-200 comes with management software, a power supply, an external battery holder, a pair of AA batteries, screwdriver, and a bag of components (refer to the DK-200 Contents document). The kit also contains three different kinds of SNAP demonstration boards, each with a SNAP Engine:

- One RF266 USB Dongle
- One SN171 ProtoBoard
- One SN172 ProtoBoard.

__Remember:__ You do not have to use the various "demo" boards in order to use the SNAP Engines in your own projects, and additional boards can be purchased from Synapse to expand your evaluation network.

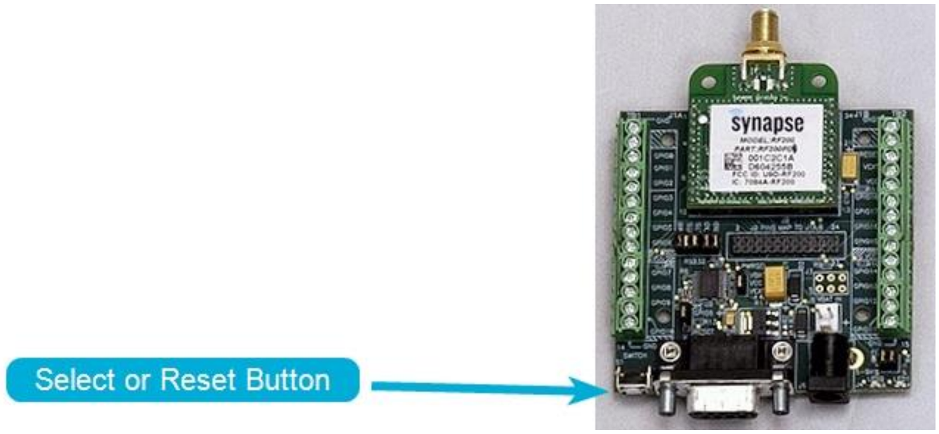
As you work through the upcoming demonstrations you will get the chance to "play" along and connect various sensors and indicators. Your kit should have come with a screwdriver, 3.2 dBm RP-SMA antenna, and bag of various electronic components.

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)

SN171 Proto Board



The Synapse SN171 Proto (prototyping) Board provides easy access to all 20 General Purpose Input/Output (GPIO) pins available on the RF200 SNAP Engine. Seven of these pins (GPIO 11-13 & 15-18) can also serve as analog inputs.

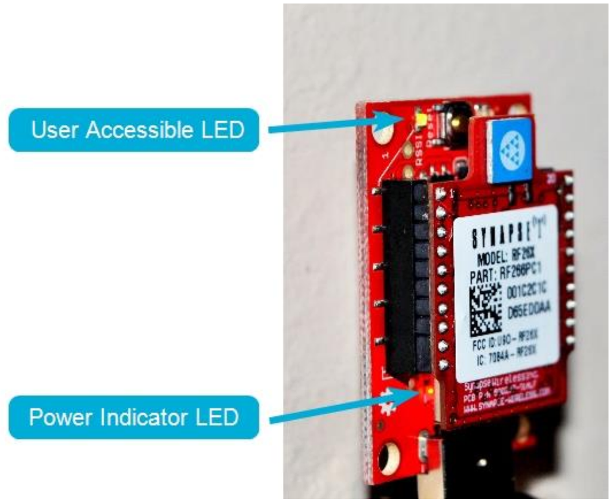
Almost all of the onboard RF Engine signals are made available at easy-access screw terminal blocks located on either side of the node. These same signals are also made available in a more compact form in the center of the board. A dual row header provides a connection point for a ribbon-cable (or some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

More information about this module's hardware can be found in the SN171 Quick Start Guide and the SNAP Hardware Technical Manual available from the Synapse website.

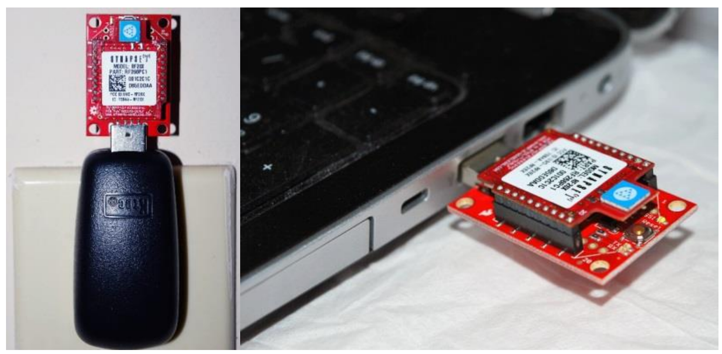
RF266 USB Dongle

The RF266 USB dongle 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 (or SNAP Connect) to a SNAP network.

A green LED is available as an output indicator. It can be controlled by manipulating SNAPpy IO number 20 via SNAPpy scripts. A second red LED is used to indicate that power is being supplied to the module. It cannot be controlled by the user. There are two more LEDs on the dongle only to indicate activity on TX and RX UART lines.



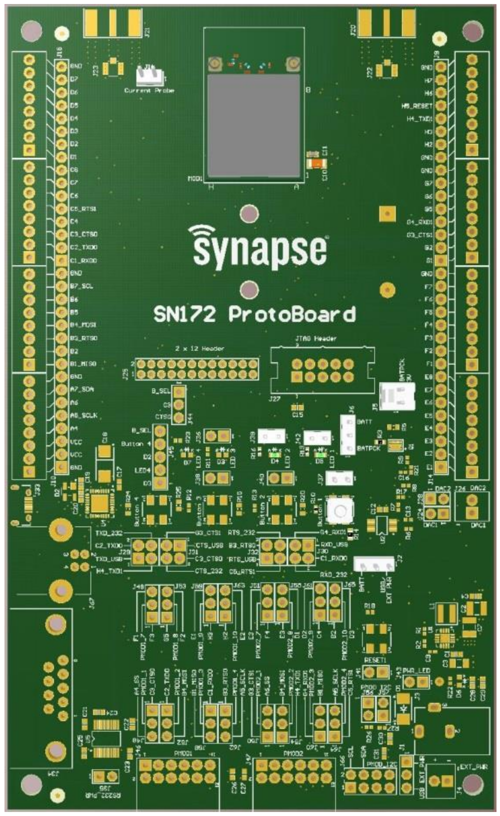
Power to this module is provided via a standard USB connection. This allows for the RF266 USB Dongle to be powered using a PC or other USB power source (such as an AC adaptor).



Note: The Portal software doesn't need to be installed for the RF266 USD Dongle to draw USB power.

More information about this module's pin configuration can be found in the SNAP Reference Manual (available from the Synapse website).

SN172 ProtoBoard



The SN172 ProtoBoard provides easy access to all 33 of the SM220 SNAP Engine's General Purpose Input/Output (GPIO) pins. Seven of these pins (A4 – A7, B4, B6 and B7) can serve as analog inputs. All the onboard RF Engine signals are available at easy-access screw terminal blocks and pin headers located on either side of the node.

> **Note**: The pin headers are not populated on Rev B of the SN172 Protoboard.

Some of these same signals are made available in a more compact form in the center of the board. A dual row header can be populated that provides a connection point for a ribbon-cable (or some other form of wiring harness) from some other circuit board or testing apparatus. This dual-row header has the same pin-out as the one on the SN171 which makes it convenient for the user to use any extension boards made according to this pin-out on both the ProtoBoards.

The SN172 can be powered by two AA batteries via an external battery holder included in the kit. More information about this module can be found in the SN172 Quick Start Guide (available on the Synapse website).

> **Note:** Connector J11 on SN172 ProtoBoard (Rev B), the purpose of which is to measure the current consumption of the SM220 RF Engine, is non-functional.

Demonstration Time

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

Each segment builds upon the concepts of the previous demonstration and they are most effective if executed in order.

RF266 USB Dongle Connection

Go ahead and connect the RF266 USB dongle to an open USB slot on the PC or a USB power adapter. If you do connect to a PC, a dialog box may appear asking if you wish to install software.

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 **Cancel** for now. For Windows Vista and later versions, the drivers should automatically be installed the first time the dongle is plugged in. If not, the drivers will get installed when Portal is installed, which is discussed later in this document.

Demonstration 1: Counting Straight Out of the Box

Both the nodes on RF266 USB Dongle and SN172 ProtoBoard come factory preloaded with a demo SNAPpy script (McastCounterDK200.py). 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 interact without controller software. They work straight out of the box.

Step 1 – Have the RF266 USB Dongle plugged into a PC or USB power supply. The module will have a single red power indicator LED lit to indicate it is powered. Keep the RF266 in a place where you can see the LEDs. Since this is the only form of (human-interpretable) output on this node, we'll use them in a couple of demonstrations.

> **Note:** Don't install the software just yet. SNAP Engines do not require that the Portal software be present to operate. Each SNAP Engine is an autonomous (and customizable, through scripting) node in a network.

Step 2 – Connect the RP-SMA antenna to the RF200 module on the SN171 ProtoBoard and power it up using the AC adapter provided with your evaluation kit. You should see the yellow LED (LED2) begin to flash. Now, install the AA batteries in the battery pack provided and power up the SN172 by connecting the battery pack to it. You should see the LED 1 begin to flash. This means we are ready to push some buttons.

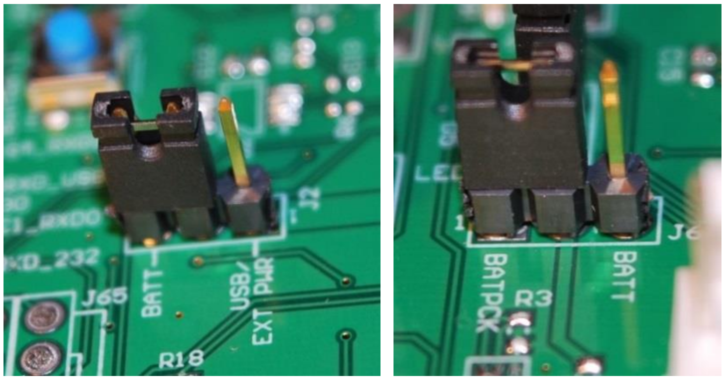
> **Note:** The RP-SMA antenna lets the modules communicate over longer distances. Please refer to the datasheet of RF200 module to get more information about the supported range.

If the nodes are on and have a lit LED, proceed to the next section. Otherwise, here are tips that may help:

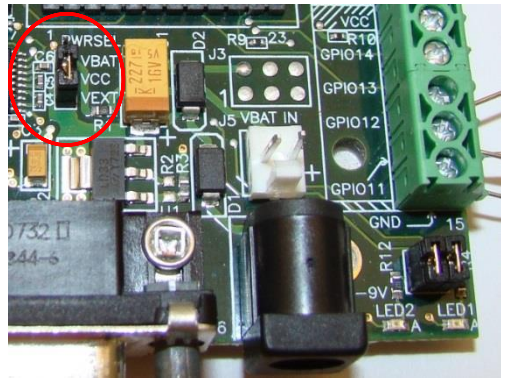
Troubleshooting

Tip #1: Verify the power related jumpers:

Your SN171 ProtoBoard should have come preconfigured to work with an external DC power supply. Verify that the PWRSEL jumper is in the VEXT position (connecting pins 2-3), and not the VBAT position (connecting pins 1-2). Your SN172 ProtoBoard should be preconfigured to work with a battery pack along with two AA batteries. This battery pack needs to be plugged into connector J5 on the board. Verify that the jumper is installed so that BATT is selected on connect J2 and the jumper is installed so that BATPCK is selected for connector J6.



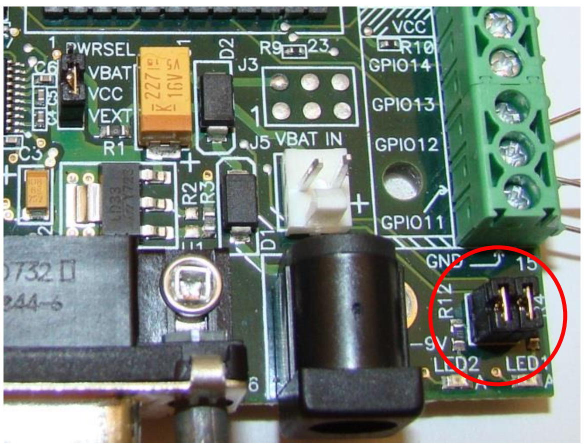
SN172



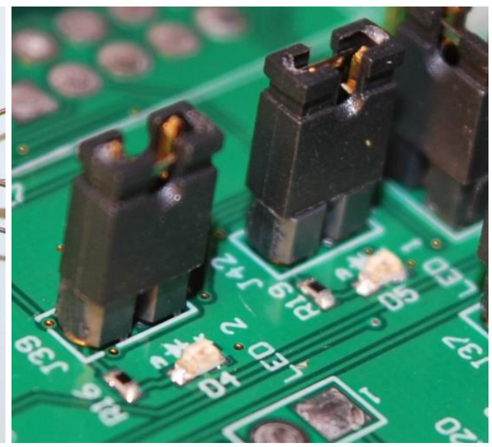
SN171

Tip #2: Verify the LED related jumpers:

The SN171 and SN172 ProtoBoards should come preconfigured to enable its two on-board LEDs. If either of the units is not blinking its LED, it is worth verifying that the LED1 and LED2 jumpers are both installed.



SN171



SN172

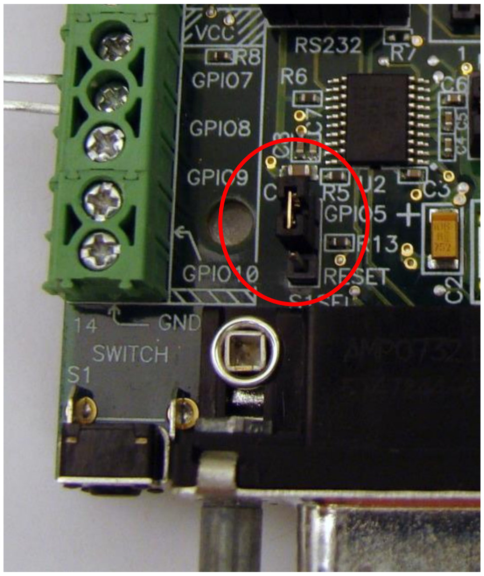
Continuing…

The pre-loaded script ("McastCounterDK200.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 either ProtoBoard (SN171 or SN172). The LEDs on the Proto Boards and the LED on the RF266 USB Dongle change pattern with each button press. Press-and-hold the button and the LED configuration will return to its original state (i.e. the count returns to zero and all the LEDs turn OFF). Press the button again and verify that the LEDs blink in the same pattern. If you power cycle the ProtoBoards, you can see that the LEDs on the ProtoBoards begin to flash again waiting for you to press the button to change the pattern.

More Troubleshooting

If you are not seeing the LEDs change on the ProtoBoards make sure that the button on them is enabled. The S1SEL jumper located up and to the right of the button on SN171 ProtoBoard should be set to 'GPIO5' (connecting pins 1 and 2) and not to 'reset' (pins 2 and 3). Verify that jumper is installed on connector J37 on SN172 ProtoBoard.



SN171

SN172

In the unlikely event that the SM220 module on SN172 ProtoBoard becomes unresponsive, you might want to perform either or all of the following to restore it to the default condition:

1. Erase SNAPpy Image
2. Factory Default NV Params

The procedure for either action can be found in Portal Reference Manual. However, the SN172 ProtoBoard needs to be serially connected to the workstation that has Portal installed on it before doing so. The simplest way to do this is to hardwire the UART1 pins on SN172 ProtoBoard to the serial cable from your workstation. The RESET pin on the SN172 ProtoBoard needs to be wiggled during the process of restoring the node to the default condition. You may use a jumper wire to wiggle the H5_RESET pin on the board.

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

- The nodes are immediately able to communicate with each other by forming a wireless mesh network. 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 is 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 have come from 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.

3. Installing Portal

The 200 Series Core IoT Development Kit contains a sheet of paper with instructions for where to download the Portal Installer and SNAP support documentation. This provides all the software you need to get started with SNAP and Portal. Future updates and releases will always be available on the Synapse support forum.

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.

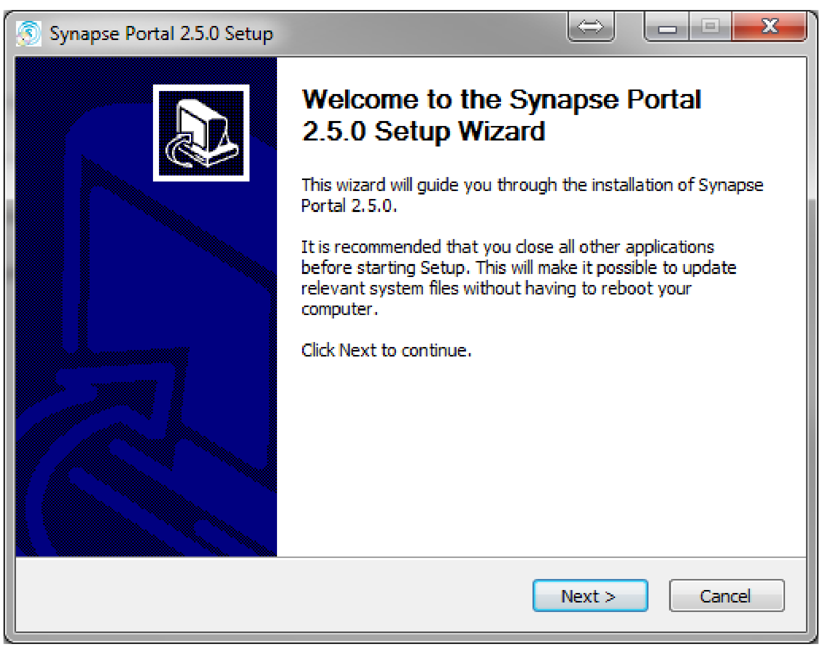
Also be sure to check the Synapse website at http://www.synapse-wireless.com

Running Setup

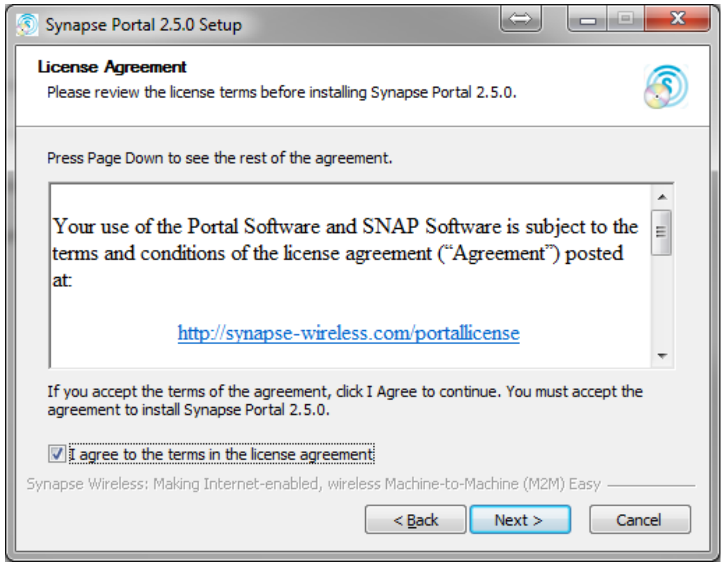
Download and run the Portal installer, Portal-setup-2.5.n.exe (where "n" indicates the latest release version). Depending on the version of Windows running on your PC, you may get a warning dialog box asking if you wish to install. The warning is harmless. Click **Yes** to proceed with the installation.

Note: If a previous version of Portal (for example, version 2.4.39) is already installed on your computer, you will get an initial dialog box asking if it is OK to remove the previous version. Click **OK** to install the newer version.

Your precise Portal version number might be different from 2.5.0, but the process should be very similar. A dialog box similar to the following will appear (your version number may be higher).

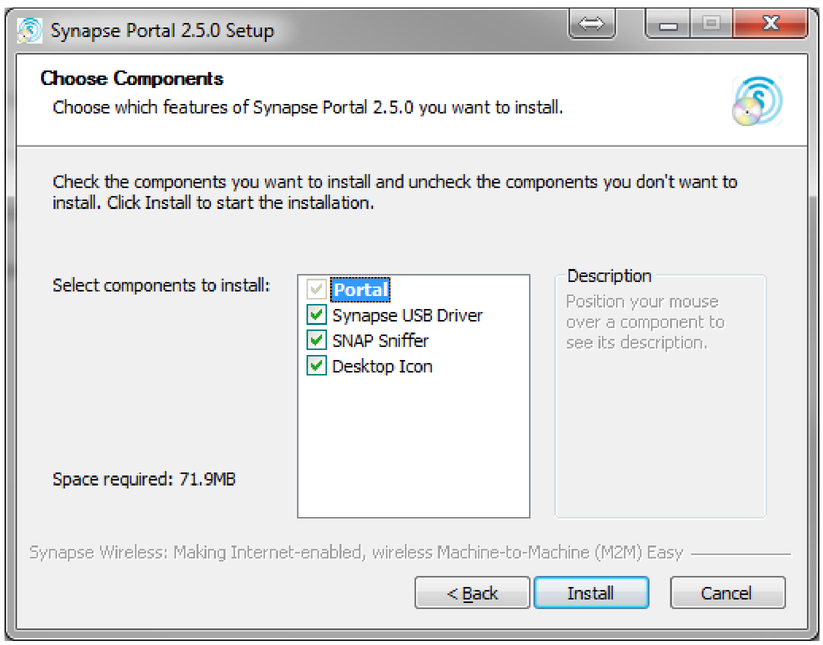


Click the **Next** button to get the following:



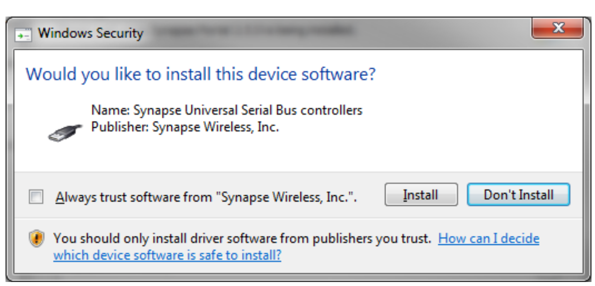
Read the license agreement at the specified URL, check the **I agree** box and then click **Next**.

Manually enter the desired destination folder, browse to the desired folder, or click **Next** to accept the default.

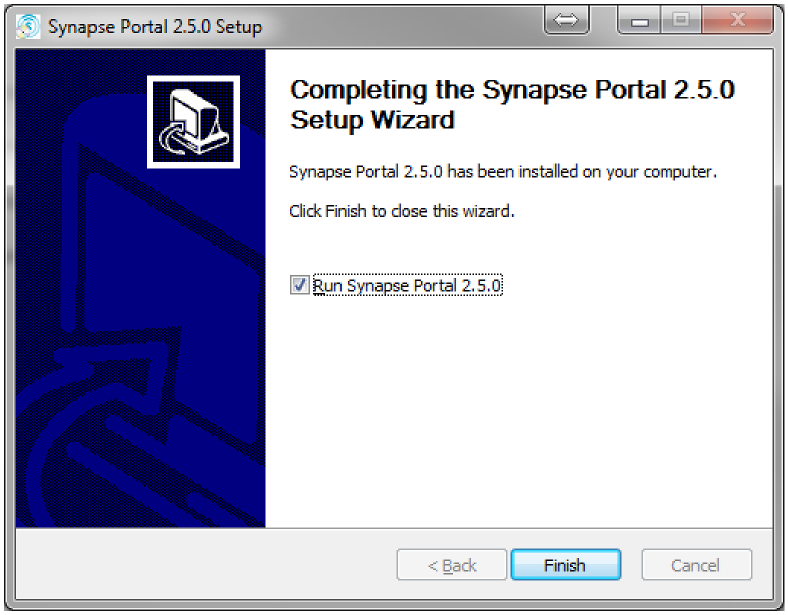


Make sure the desired components are checked, and click **Install**.

After several files have been processed, if you specified that USB drivers should be installed 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 the **Install** button. The installation process will continue.



You have now successfully installed Portal. There will be a Portal icon on your Windows Desktop (if you specified there should be), as well as in the Start Menu. We recommend that you do not run Portal until you have completed the bridge device driver installation through the following steps, so you should uncheck the **Run Synapse Portal** checkbox before clicking **Finish**.

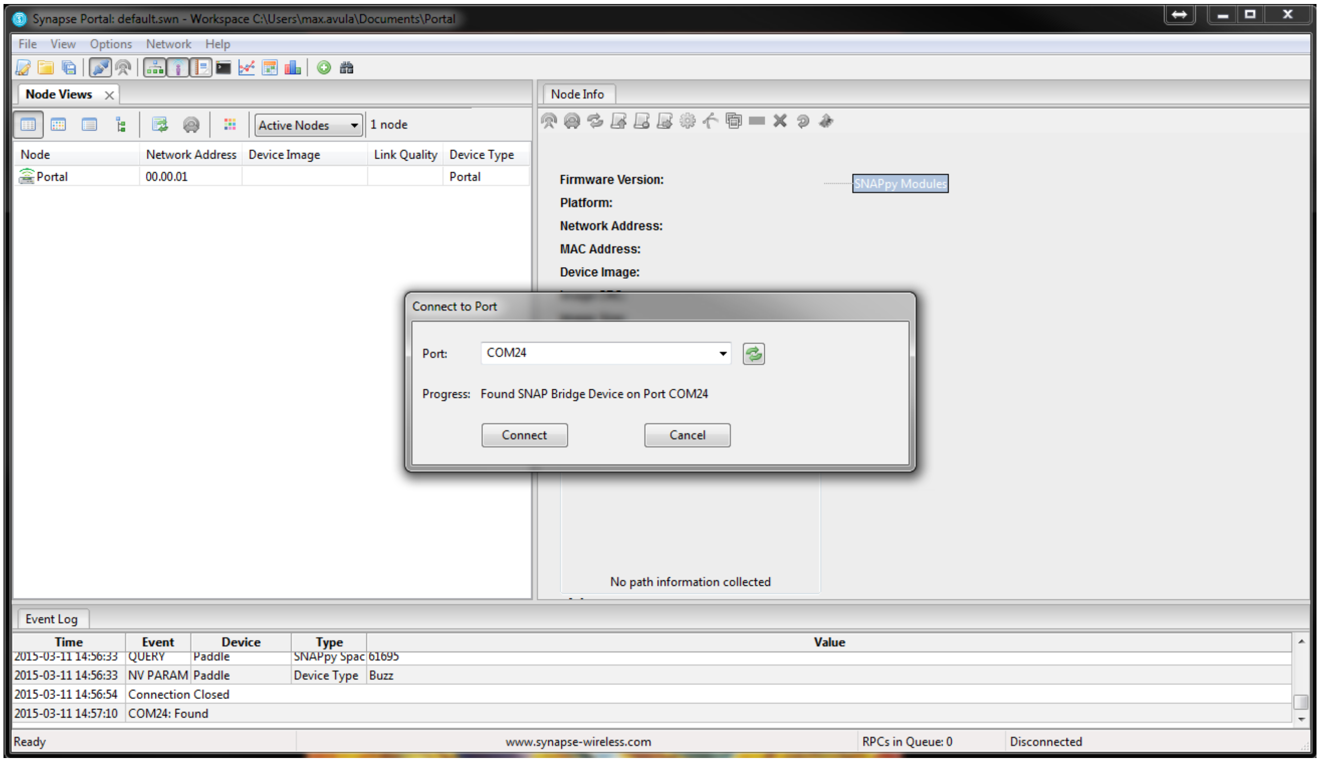
Plug in the Bridge Device

The RF266 USB Dongle should now be plugged into a USB port on the PC on which you installed Portal. Depending on the drivers loaded on your system and which version of Windows you are using, it may be necessary for the Synapse USB drivers to complete installation. This will only occur the first time the bridge device is connected and powered up. For Windows 7 or higher, the drivers for RF266 USB Dongle are automatically downloaded the first time it is plugged in (as shown earlier).

Note: If the drivers for the USB dongle are not automatically installed when plugged in for the first time, please download and install the latest FTDI driver from the website http://www.ftdichip.com/Drivers/VCP.htm

Program Launch

After installation, launch the Portal program. You should see a screen similar to the following:



If you do see something similar, you have now successfully installed Portal and detected a bridge device connected to USB serial port. By default, the "Connect" dialog box is automatically shown at Portal startup. If you click the **Cancel** button, you can bring this dialog back by clicking the  button on the main toolbar.

Also be aware that this toolbar button doubles as a status indicator. When you are connected, it looks like , and functions as a disconnect button. When you are not connected, it looks like  and functions as a connect button.

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 RF266 USB Dongle will act as the bridge.

If you have an RS232 connection on your PC, you can connect to the RF200 on the SN171 ProtoBoard instead.

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 RF266 USB Dongle from the previous section, so keep it connected to the COM port (USB Serial).

Press **Connect** once the bridge device (RF266 USB Dongle) has been detected (this is probably going to be COMx, where 'x' is usually 1, 2, 3, and so on).

Navigating within Portal

Portal is straightforward to use. However, since it is extremely customizable by the user, your screen layout may not always 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** operations.

Clicking one of these top-level menu choices will pull down a sub-menu of additional choices. For example, clicking **Network** will present a sub-menu from which you perform actions like **Broadcast Ping**, **Find Nodes…**, or **New Configuration**. Similarly, clicking **Help** will bring up choices for **SNAP Reference Manual**, **Portal Reference Manual**, etc.

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 cursor over each button will display a short "tool-tip" help message, and clicking each button will initiate the action displayed by the tool-tip.

Tabbed Windows

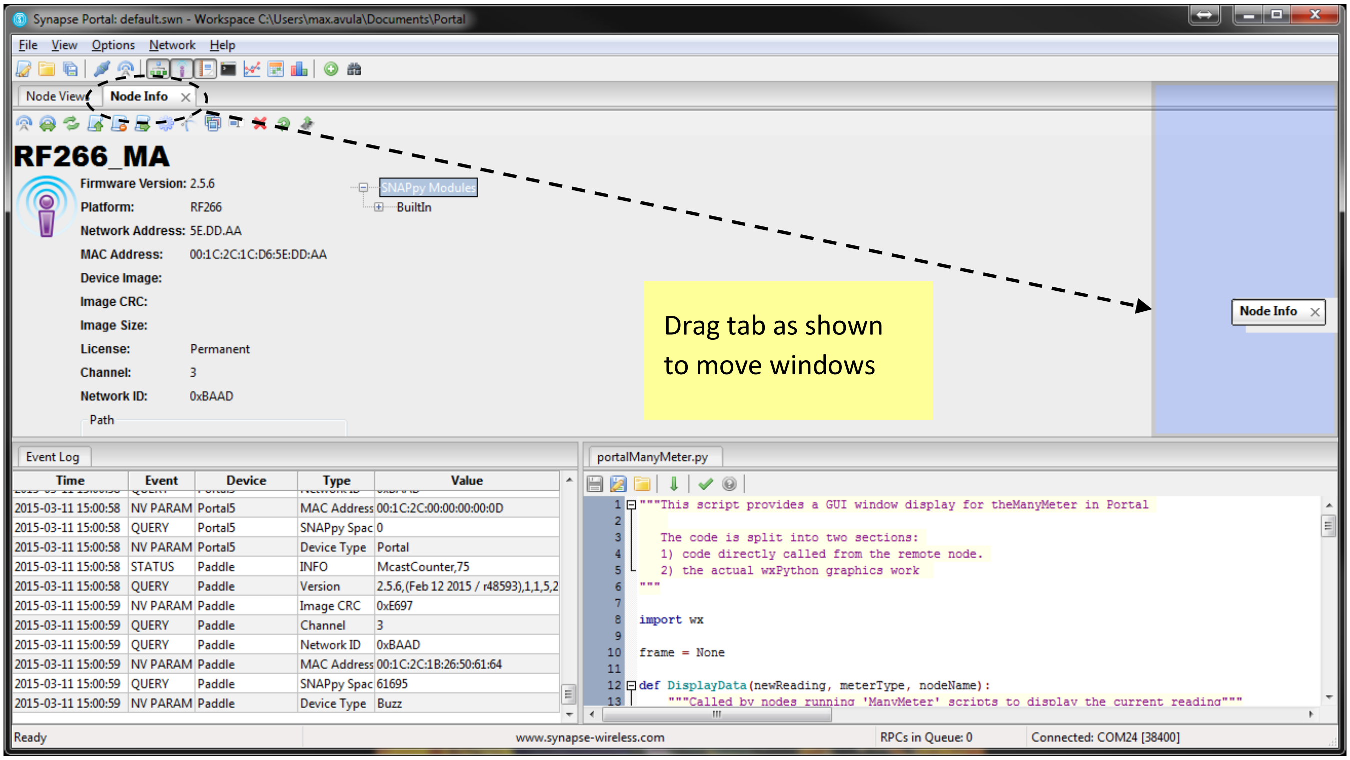
The remainder of the Portal GUI is taken up by a changeable collection of tabbed windows. Each of these windows has a name, which is displayed in the tab for that window.

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

Portal starts out with an initial set of tabbed windows visible. Sometimes clicking 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 the small **X** located to the right of the name in the tab when the tab has focus.

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 **Views**, then choose **Node Views window**. Alternatively, you can click 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 multiple ways:

- **Report View**

- Icon View
- List View
- Tree View

All four views are just that, “views” of the same network information.

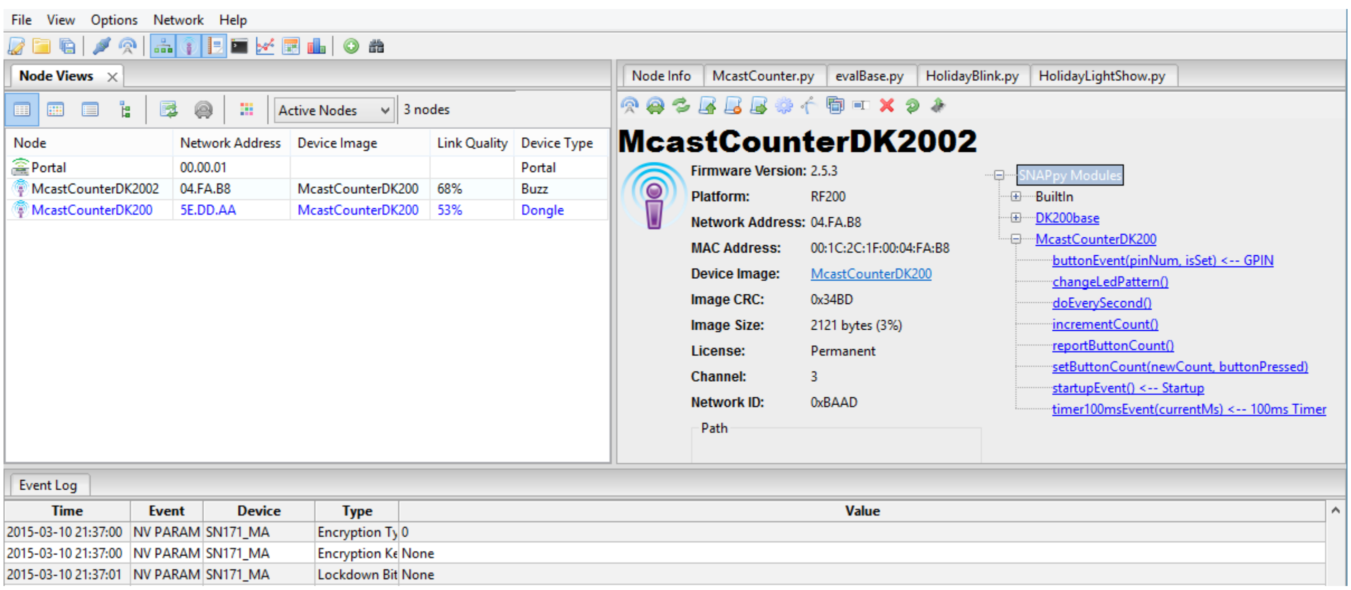
Note: If a node isn’t showing up in the list, you can click the Ping button to refresh it.

Click the **Report View** button.

You should see that three Devices have been discovered. They all will have names of the form McastCounterDK200X. The directly connected “bridge” device (the RF266 USB Dongle) should be shown in blue, while the remote devices should be displayed in black. Because the nodes report in using a random response delay, which node gets to be McastCounterDK200 and which node gets to be McastCounterDK2002 can vary.

Note: Although unlikely, if any of the nodes are not showing up on the list, you might have to restore it to the default settings using Portal. Information on how to restore a node to its default condition can be found in the Portal Reference Manual. Please note that, for SN172 ProtoBoard, you have to establish a serial connection manually by hardwiring to the corresponding UART 1 pins on the board.

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



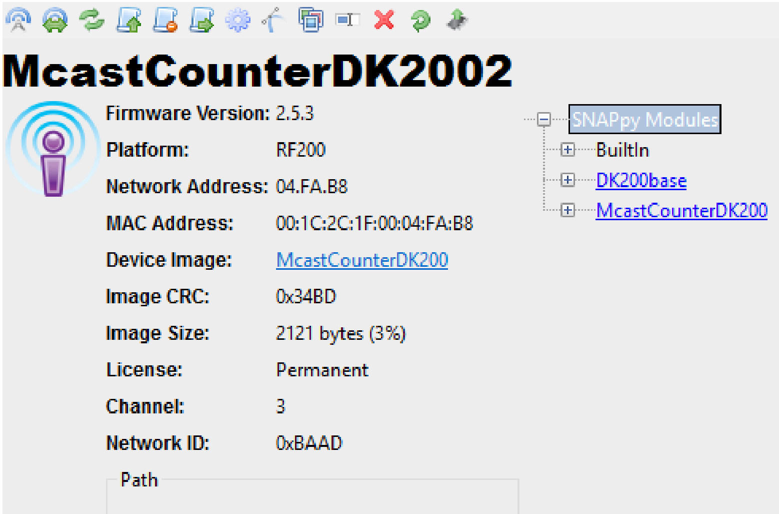
Note that the node names are based on the names of the SNAPpy scripts loaded into those same nodes at the time of discovery, but with additional trailing digits (2, 3, 4, etc..) added to enforce uniqueness.

Since these nodes were pre-loaded at the factory with the “McastCounterDK200.py” script, their names are of the form McastCounterDK200X, 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”, “Node1” and “Node2”.

Node Info

Lots of information is shown in the **Node Info** tabbed window. However, Portal may not be set to automatically query the node for its information. (This is a configurable preference in Portal.) To be sure Portal knows everything important about your node, click the **Refresh Node Information** icon in the toolbar that runs across the top of the **Node Info** tab.

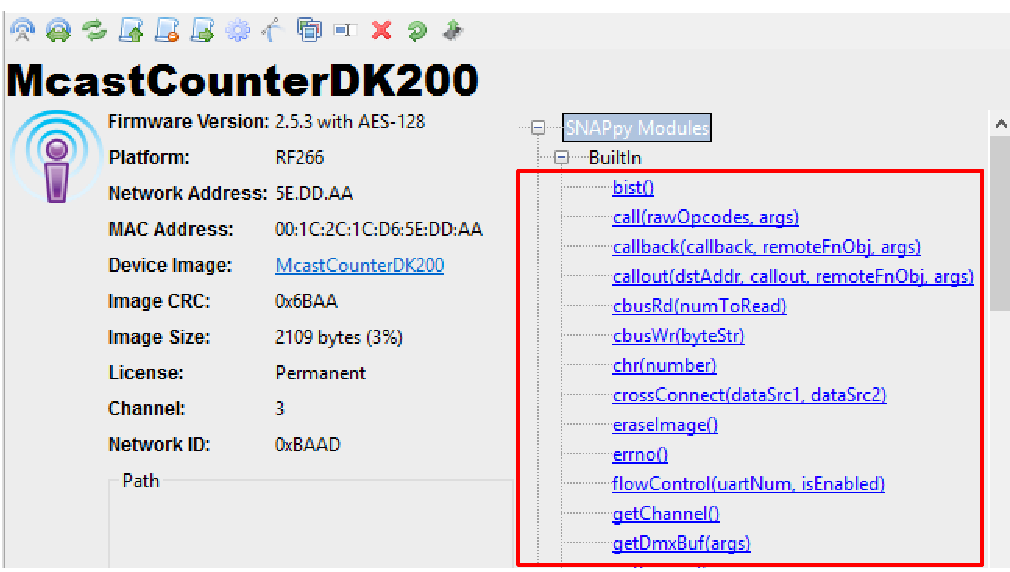


Across the top, a toolbar provides easy access to node-specific functions.

On the left-hand side, the **Firmware Version**, **Platform**, **Network Address**, **MAC Address**, **Device Image**, **Image CRC**, **Image Size**, **License**, **Channel**, and **Network Id** are shown. Below that is a block where **Path** information (the path to/from the node) can be displayed. Below that there is an **Info** field that 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 (also referred to as a SNAPpy image) loaded into the node. Here you can see that the script/image “McastCounterDK200.py” has been loaded into the node. You can click the device image name shown (McastCounterDK200), and automatically bring that script up in Portal’s built-in source 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 cursor over a function name will display a tool-tip for that function. More importantly, you can click 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.

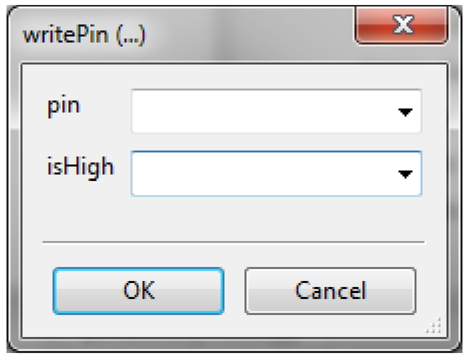
If the function requires any parameters, Portal will automatically prompt you for them.

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

You can either:

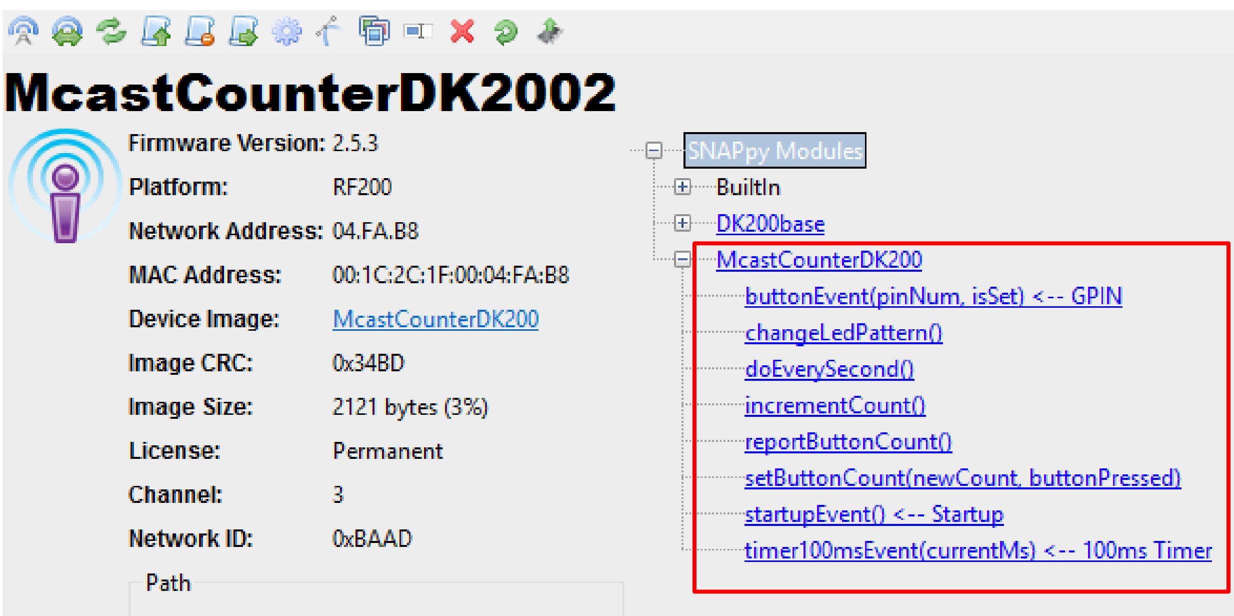
1) Enter a value (ex. pin = 1, isHigh = True) and press OK, or

2) You can press Cancel to abort the function invocation.



Don't forget that in the Python language 'True' and 'False' are case sensitive (with the 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 "McastCounterDK200.py" SNAPpy script.

Like the built-in functions, these can also be directly invoked by clicking 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.

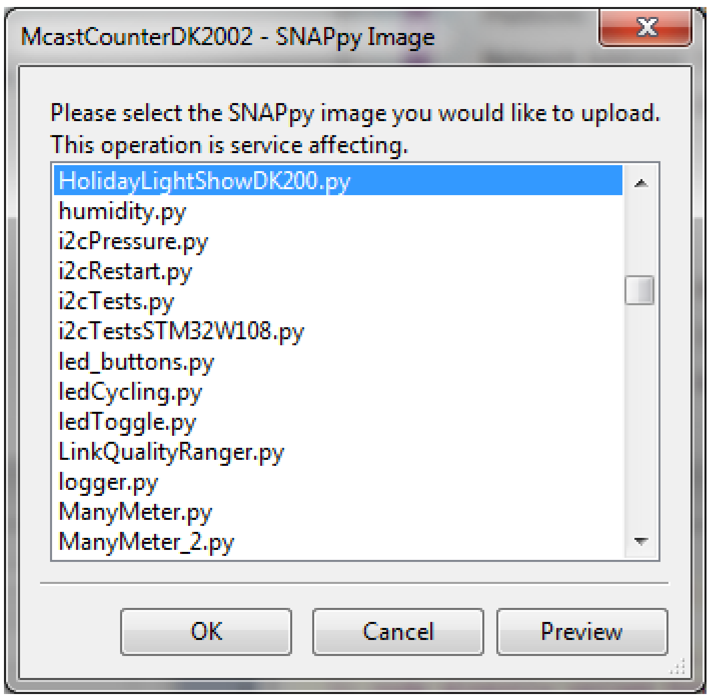
We've already tried out the "multicast counter" functionality in the first demonstration, so now let's override that behavior with some different ones. To do that, we will assign new "behavior" by giving each node a new SNAPpy Image.

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" . (If the button is disabled, it means Portal does not know enough about the node to enable it to load a script into it. Click the **Refresh Node Information** button, and after the node information refreshes the **Upload SNAPpy Image** button should be enabled.)

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 "HolidayLightShowDK200.py" example, and click **OK**. Upload of the new SNAPpy Image (over the air!) should complete in a few seconds. The node will automatically restart when the upload finishes.

Next select the node with the device type set to "Dongle" from the Node List and upload the "HolidayBlinkDK200.py" script into it. Also, upload "HolidayBlinkDK200.py" to the node with the device type set to "Proto".



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, we 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.

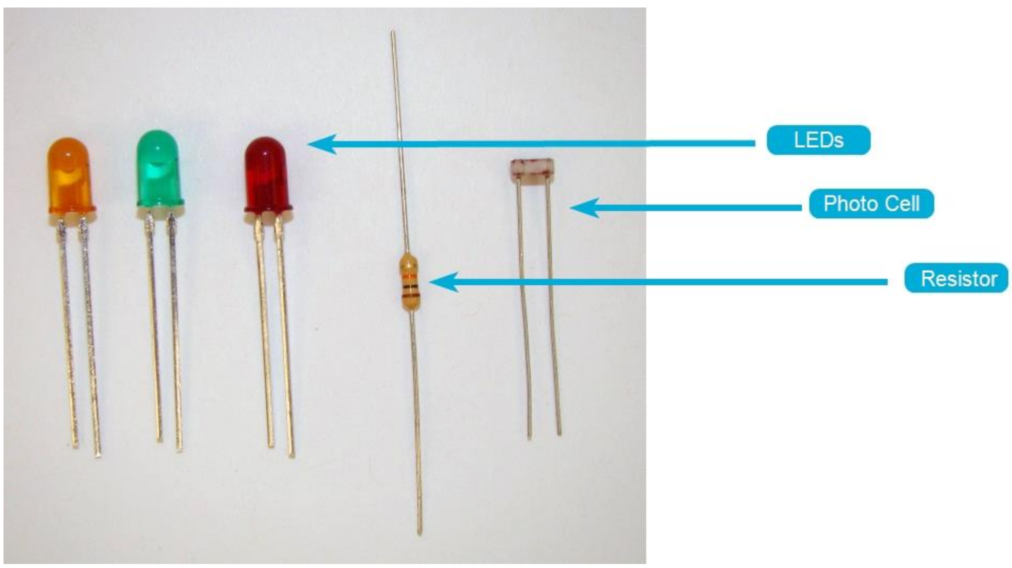
Demonstration 2: Holiday Light Show

You should now have the RF266 USB Dongle and SN172 ProtoBoard nodes running the example script "HolidayBlinkDK200" and the Proto Board node running another example script "HolidayLightShowDK200". 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 (the beige one)
- 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 loosens the connector terminals.

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

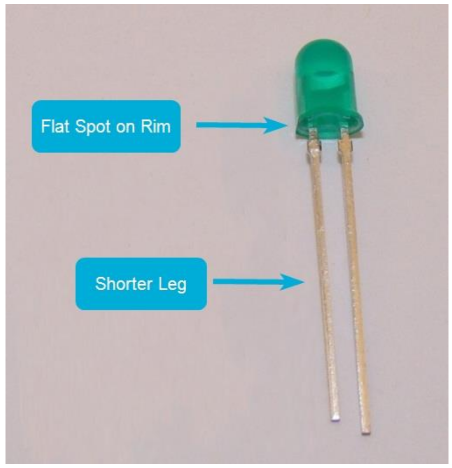
Note: Resistors have color bands on them that indicate how much resistance they provide. But rather than providing a lesson in color bands, we have provided resistors with different body colors, too. The 10K Ohm part is beige in color, while the 100K Ohm resistor is blue. We'll use the beige resistor 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 SN171 ProtoBoard.

Step 2 – Determining LED polarity:

Find the green LED. LEDs do have a polarity, meaning they can only be connected in one way, 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:

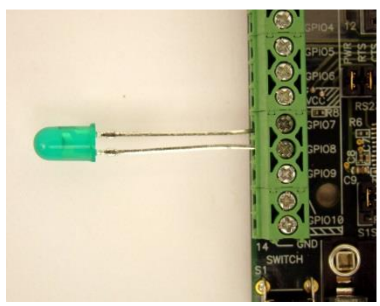


Step 3 – Connecting the green LED:

Place the shorter "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 — note that it is NOT next to GPIO 7, as there is a VCC connection between them).

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 the legs of the LED. (See picture.)

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

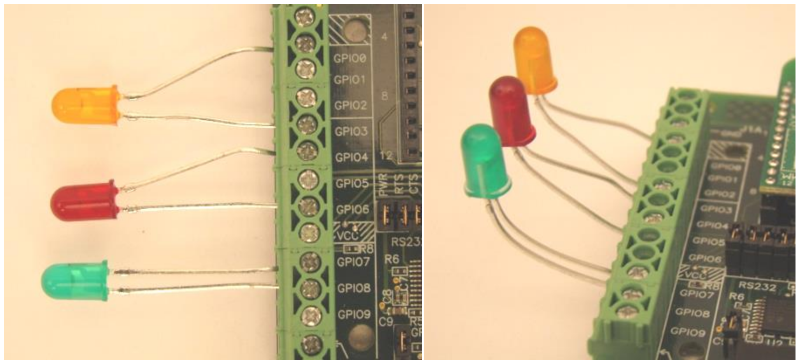
Step 5 – Connecting the amber LED:

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.



Step 6 – 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.)

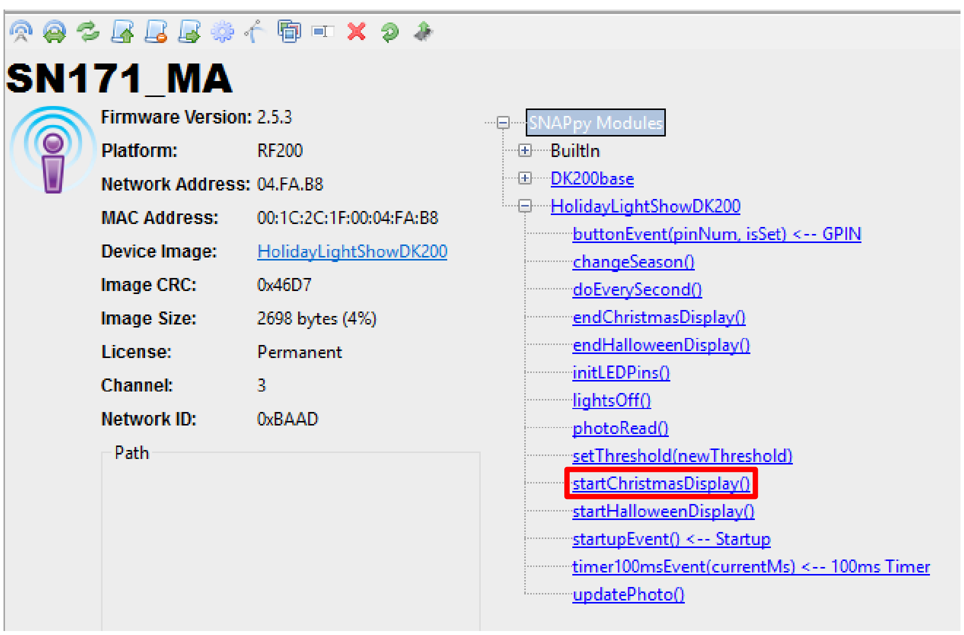
It's time to check on the quality of our work by testing the LEDs. We can use the new SNAPpy script we just uploaded to the node for this purpose.

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 SN171 ProtoBoard, which has its Device Type set to "Buzz". (It will be running the "HolidayLightShowDK200" script.)

Switch to the **Node Info** pane and find the "HolidayLightShowDK200" specific section from the list of available functions. You might need to collapse the "DK200base" section of functions, or to scroll down to find the right functions.



Click the startChristmasDisplay() function.

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

Click the changeSeason() function.

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

Click the lightsOff() function.

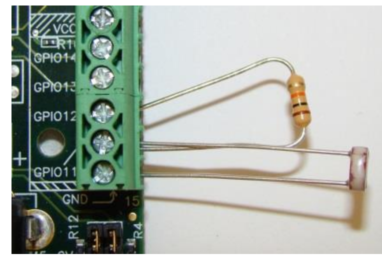
All 3 LEDs should now be off (we are ignoring the SN171 ProtoBoard's built-in yellow LED).

If you did not see each of the LEDs flash at some point during the test, check your connections. Try a gentle tug on each LED lead. 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?*

Step 7 – 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.

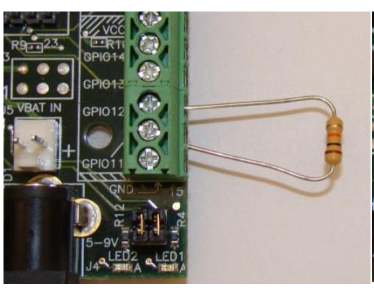
Step 8 – Connecting the photo-cell:

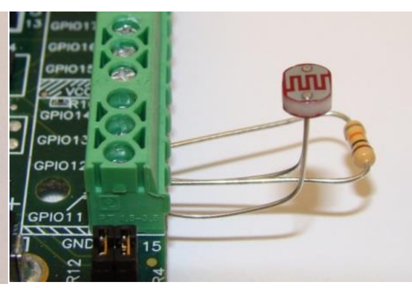
Place one leg of the photo-cell into the pin labeled GPIO 11 (don't worry; it is supposed to share the connector with the resistor) and the other leg in the last slot in the connector (the GND pin). There is no polarity on the Photo cell, so it does not matter which leg you choose for which pin.

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 'neater' setup.





> **Note**: While not used in any demo scripts, there are software-controlled internal pull-up resistors available 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 "HolidayLightShowDK200" 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 RF266 USB Dongle and SN172 ProtoBoard. You will notice that the LED on RF266 toggles every three quarters of a second and the two LEDs on SN172 ProtoBoard are toggling as well. The SNAP node on the SN171 ProtoBoard has communicated the change in sensor status over the air to the neighboring devices to extend 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 or changes the light show using the node's setThreshold() function in Portal. (The default value 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 SN171 ProtoBoard has wirelessly communicated the message to change the light pattern to the RF266 USB Dongle and SN172 ProtoBoard as well. The LED on RF266 will toggle every half a second whereas the LEDs on SN172 will toggle in a pattern opposite to that in step 11.

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. This will send a message to both the RF266 USB Dongle and SN172 ProtoBoard to turn off their respective LEDs.

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:

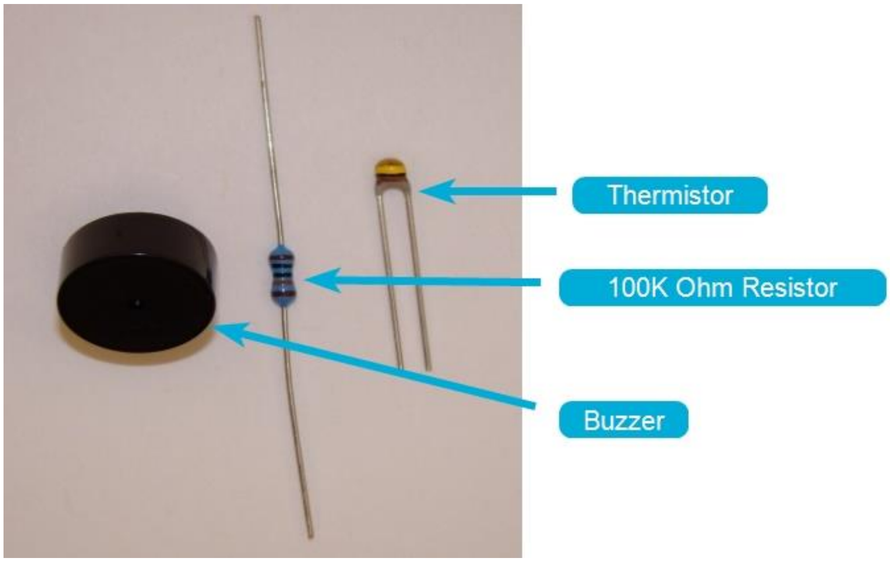
If you peek at the HolidayLightShowDK200 SNAPpy script you will see how easy it is for SNAP nodes to communicate. It contains a line of code: "mcastRpc(1,2,"christmasBlink")". This single line is all it takes for a SNAP node to send a message to a group of devices.

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 SN171 ProtoBoard 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:



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

Remember, just as with a standard screw, clockwise tightens and counter-clockwise loosens the connector terminals.

We'll use the pins in the upper right hand corner of the SN171 ProtoBoard (GPIO 18, VCC, and GND) to attach the thermistor and the 100K pull-up resistor for this demonstration.

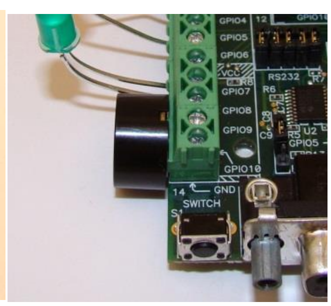
Note: You can differentiate the included resistors by color. The 10K Ohm part is beige in color, while the 100K Ohm resistor is blue. We'll add the blue resistor for this demonstration.

Step 1 – Disconnect the power running to the SN171 ProtoBoard.

Step 2 – Connecting the buzzer to the SN171 ProtoBoard:

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:

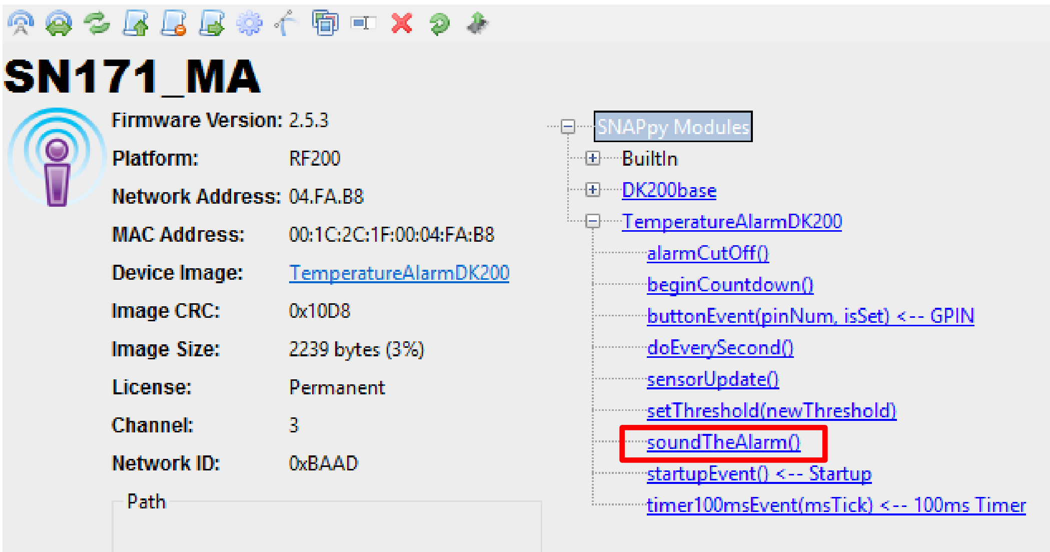
Connect the power back to the SN171 ProtoBoard.

Load the script “TemperatureAlarmBridgeDK200.py” onto the RF266 USB Dongle (i.e. the unit with the Device Type set to “Dongle”) using the step you learned in the “Uploading SNAPpy scripts” section.

Load the script “TemperatureAlarmDK200.py” onto the SN171 ProtoBoard (i.e. the unit with the Device Type set to “Buzz”).

Step 4 – Make sure that you still have the SN171 ProtoBoard’s node (the one with “TemperatureAlarm” in it) selected in the **Node View** pane.

Switch to the **Node Info** pane and find the “TemperatureAlarmDK200” specific section from the list of available functions.

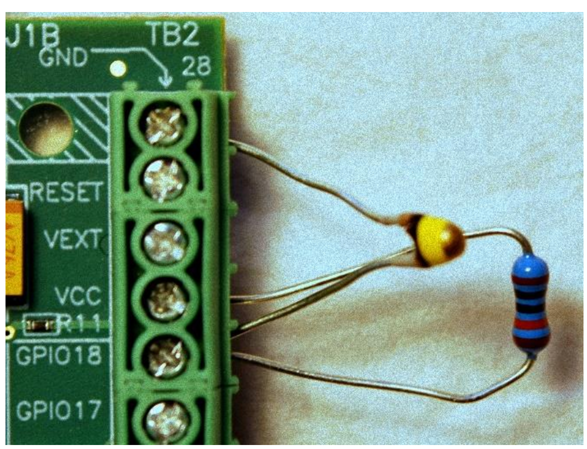


Click the soundAlarm() function to test the buzzer. You should hear a long beep from the buzzer. (If you do not hear the beep, confirm that you have the buzzer’s polarity correct in your installation.)

Disconnect the power running to the SN171 ProtoBoard again.

Step 5 – Connecting the resistor (blue) to the SN171 ProtoBoard:

Bend the legs of the resistor toward one another as seen in the picture



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

Place the other leg into the connector at the pin labeled VCC (located on the upper right-hand side). There is no polarity on this resistor, so it does not matter which leg you choose for which pin. Note that there are two terminals labeled VCC. You could use either of them here, but for clarity we recommend you use the one next to GPIO 18.

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?)

Step 6 – Connecting the thermistor to the Proto Board:

Place one leg of the thermistor (see picture for identification) into the pin labeled GPIO18 (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 above 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 SN171 Proto Board.

Step 8 – Hold the thermistor between your thumb and forefinger to raise the temperature. (If you have especially cold hands, it may be necessary to warm your hands first, or to rub the thermistor to warm it up. You may also blow over the thermistor to warm it up.) 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 SN171 ProtoBoard and the flashing LED on the RF266 USB Dongle.

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 "TemperatureAlarmDK200" specific section of the **Node Info** pane within Portal (this can be done on either node).
2. By pressing the built-in SN171 ProtoBoard button.

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

KEY POINTS:

- Synapse RF engines have the ability to read analog inputs.
    - RF200 SNAP Engines have 7 analog inputs included in the 20 GPIO pins. Other platforms of SNAP Engines have different amounts based on their underlying processor hardware.
- 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

If you are interested in the types of things SNAP Engines can do, you have probably heard of a 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 "ManyMeterDK200.py" SNAPpy script onto the SN171 ProtoBoard (a skill we've mastered from previous sections, but here is a hint:).

Let's give the SN171 ProtoBoard a name. Click the **Change Configuration** icon from the toolbar and then select the Device tab. Change the Device Name (currently "Buzz" or blank) to read "OurDemo". Click "OK".

Step 2 – Switch back to the **Node View** pane and select the node associated with the RF266 USB Dongle.

Switch to the **Node Info** pane and Erase the script currently running on the RF266 USB Dongle by clicking the icon located in the toolbar.

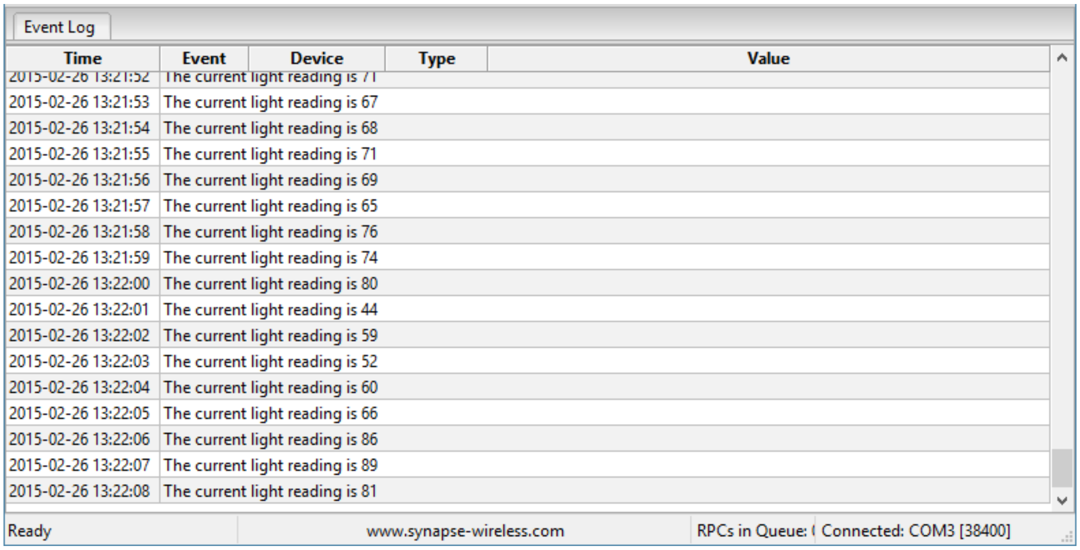
Select the **New Configuration…** option from the **Network** pull-down menu in Portal. This will reset your view to use our new name. The RF266 USB Dongle without a script will now have a node name like "Node" or "Node2". (This assumes you have not changed any Portal preferences from the defaults with which Portal was installed. If you are working with an existing Portal installation where someone may have adjusted the Portal preferences, it may be necessary to click the **Broadcast Ping** icon in the Portal toolbar to make the nodes reappear.)

> **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, if not already open. You'll see that the remote node (the one on the SN171 ProtoBoard) 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.

> **Note:** If the reading is not appearing in the event-log pane, select the node and, within the **Node Info** pane, make sure that the **Intercept Node Output** button is clicked.

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



Step 4 – Push the button located on the SN171 ProtoBoard 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.)

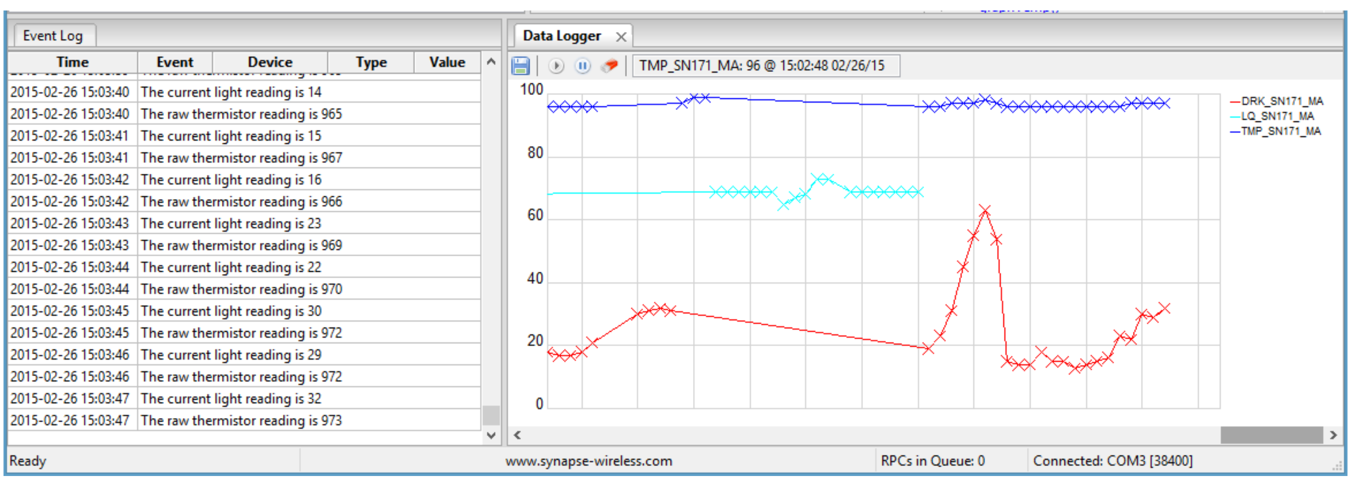
Push it again and the radio link quality will be displayed.

Push it one more time and to 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 **Save** 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 SN171 ProtoBoard to cycle through each available sensor reading and watch Portal graph the 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 "ManyMeterDK200.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:

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

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

The fact that 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.

The following 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 (i.e. the devices containing SNAP Engines). However, the Portal software is also a node in the SNAP network, using the connected bridge node to send and receive network traffic.

Step 1 –Go to the **Node View** in Portal. Select the Portal node and upload the "portalManyMeterDK200.py" script using the **Change Portal Base File** () button. The file will be located at "My Documents\Portal". [1]

Step 2 – From the same **Node View**, verify that the SN171 ProtoBoard (node with the Device Name set to "OurDemo") is still running the "ManyMeterDK200" script. Click 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 for opening the script is to use the **Open File…** command from the File menu.)

Step 3 – Edit the SNAPpy script:

The original script is read-only. So, Click 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 have cause to use it now, so let's un-comment it.

[1] Portal scripts are not, by default, in the same directory as SNAPpy scripts. Portal is capable of loading any Python script you wish to run, so you can navigate to any location on your system to load a script into Portal, while SNAPpy scripts are selected from a specific directory.

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:

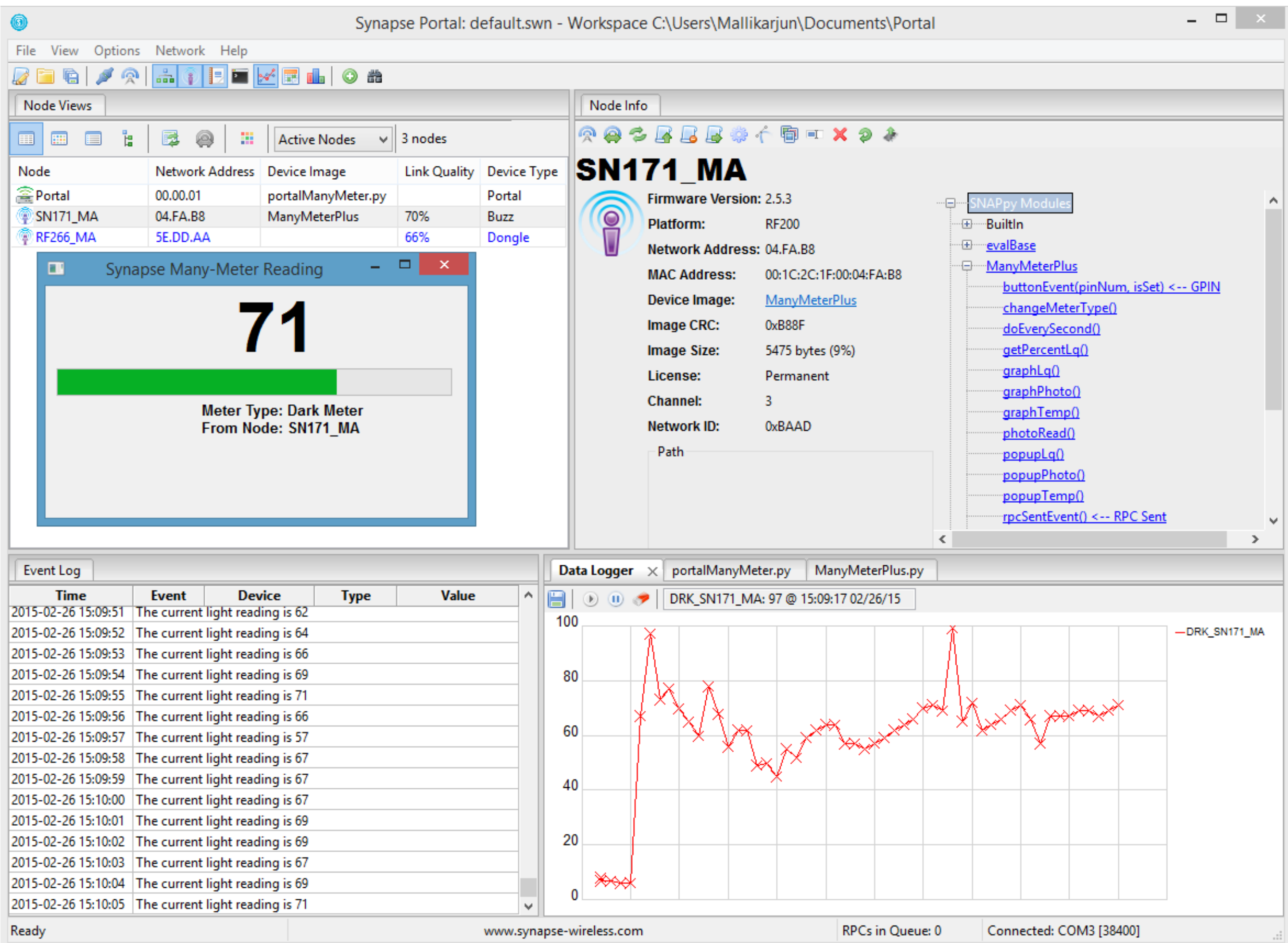
```
# EDIT NEXT LINE: This is special code to call into the wxPython functionality of Portal
rpc(portalAddr,"DisplayData",photoVal,"Dark Meter",loadNvParam(NV_DEVICE_NAME_ID))
```

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

Save your work by clicking 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 on your computer screen 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 for a second or two.



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, just as other SNAP nodes can.
- SNAPpy 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 send email to different addresses based on observed events and conditions. More details are available within the Portal Reference Manual.

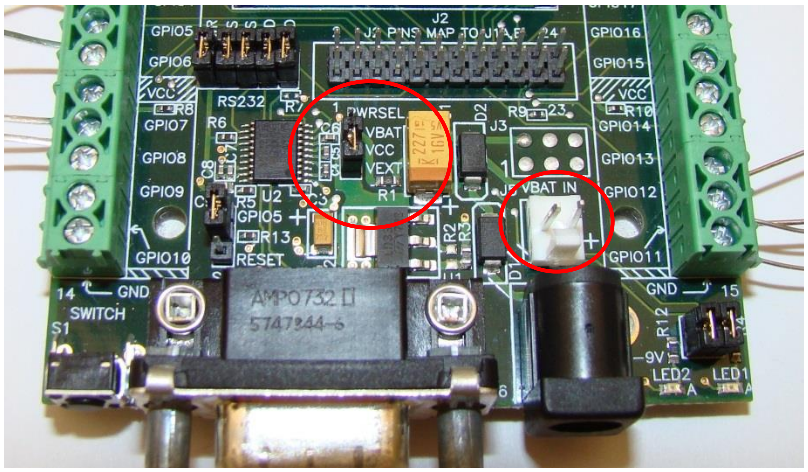
6. Alternative Energy Settings

Battery Operation

So far, we have been running the nodes from USB power (the RF266 USB Dongle), wall power (the SN171 Proto Board) and from battery pack (the SN172 ProtoBoard).

It is also possible to run the SN171 Proto Board from battery power using the battery pack included in the DK-200. 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.

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

A Snap Engine on the SN171 or SN172 ProtoBoard is capable of achieving years of battery life from the included AA pack, but to do so requires running a SNAPpy script that sleeps, as well as removing all RS232 jumpers (JMP2, JMP5, JMP6, JMP7 and JMP8) from the SN171 ProtoBoard.

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 “protoSleepCasterDK200.py” for one example of low-power operation. This script is like McastCounterDK200.py, but sleeps between button presses.

7. Where To Go Next

In this manual we have introduced the components of the DK-200, installed Portal, and run some simple demos.

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

- The “SNAP Users Guide”
- The “SNAP Reference Manual”
- The “Portal Reference Manual”
- The “SNAP Hardware Technical Manual”
- The “SN171 ProtoBoard Quick Start Guide”
- The “SN172 ProtoBoard Quick Start Guide”

These documents are in Portable Document Format (PDF) files and are available from the Synapse Wireless support forum website. The SNAP Reference Manual and Portal Reference Manual (plus other documentation) are also available from the Help menu in Portal.

Want more?

Details regarding other hardware can be found online at: http://www.synapse-wireless.com

You can also purchase these boards individually (outside of kit form) as additional nodes. They will interact with any of the included DK-200 demonstration boards.

The DK-200 only includes three types of Synapse SNAP Engines: RF200, RF266 and SM220. Other forms of SNAP Engines that may not have been mentioned in this document can also be purchased on an individual basis.

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