



USERS GUIDE

EK2100-220

Evaluation Kit

Document Revision v1.0

©2015 Synapse, All Rights Reserved. All Synapse products are patent pending. Synapse, the Synapse logo, **SNAP**, and **Portal** are all registered trademarks of Synapse Wireless, Inc.

Doc# 116-101520-015-A000

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

Disclaimers

This evaluation board/kit is intended for use for ENGINEERING DEVELOPMENT, DEMONSTRATION, OR EVALUATION PURPOSES ONLY and is not considered by Synapse to be a finished end-product fit for general consumer use. Persons handling the product(s) must have electronics training and observe good engineering practice standards. As such, the goods being provided are not intended to be complete in terms of required design, marketing, and/or manufacturing-related protective considerations, including product safety and environmental measures typically found in end products that incorporate such semiconductor components or circuit boards. This evaluation board/kit does not fall within the scope of the European Union directives regarding electromagnetic compatibility, restricted substances (RoHS), recycling (WEEE), FCC, CE or UL, and therefore may not meet the technical requirements of these directives or other related directives.

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. Errata 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:

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

Redistribution of code and use in source and binary forms, with or without modification, are permitted provided that it retains the copyright notice, operates only on **SNAP**® networks, and the paragraphs below in the documentation and/or other materials are provided with the distribution:

Copyright 2008-2015, Synapse Wireless Inc., All rights Reserved.

Neither the name of Synapse nor the names of contributors may be used to endorse or promote products derived from this software without specific prior written permission.

This software is provided "AS IS," without a warranty of any kind. ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE HEREBY EXCLUDED. SYNAPSE AND ITS LICENSORS SHALL NOT BE LIABLE FOR ANY DAMAGES SUFFERED BY LICENSEE AS A RESULT OF USING, MODIFYING OR DISTRIBUTING THIS SOFTWARE OR ITS DERIVATIVES. IN NO EVENT WILL SYNAPSE OR ITS LICENSORS BE LIABLE FOR ANY LOST REVENUE, PROFIT OR DATA, OR FOR DIRECT, INDIRECT, SPECIAL, CONSEQUENTIAL, INCIDENTAL OR PUNITIVE DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF THE USE OF OR INABILITY TO USE THIS SOFTWARE, EVEN IF SYNAPSE HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Table of Contents

Introduction	1
Getting Started	4
A Portal into Your Network	5
Evaluation Kit Hardware	6
Demonstration 1: Counting Straight Out of the Box	8
Wait! It didn't work	9
Installing Portal	11
Launching Portal	12
Using Portal	13
Navigating within Portal	13
Discovery	14
Node Info	15
Uploading SNAPpy Images	16
Demonstration 2: Holiday Light Show	17
Wait! It didn't work	21
Demonstration 3: Temperature Alarm	22
Wait! It didn't work	24
Portal's Extended Capabilities	25
Built-in Functionality	25
Demonstration 4: The Many-Meter	25
Advanced Functionality:	27
Demonstration 4b: The Many-Meter extended	27
Alternative Energy Settings	29
Battery Operation	29
Low Power Operation	29
Where To Go Next	30

Introduction

Perfect device connectivity isn't just the ability to collect data from a device or respond to that data. Perfect connectivity is knowing, responding, and evolving; and doing it all efficiently. Synapse enabled devices communicate with users and each other to form a holistic view of previously inaccessible information.

By creating a way to easily connect computers to the tangible world, people become empowered to solve problems in new ways, and create new solutions, new markets, and new opportunities. The future of connected devices isn't the web.

It's the ability to extend your senses.

Synapse technology unites the "things" in the "Internet of Things" (IoT) to create a unified sense-and-respond network that knows where to collect data, when to send alerts, and how to automatically respond. Users are empowered to cost effectively solve problems faster than before, and intelligent systems can identify, and possibly resolve, many problems before they start. The network becomes less a communications medium, and more like a nervous system.

Synapse Wireless gives you the tools and the insights to extend your senses into the digital world of your devices.

Synapse approaches device connectivity from multiple levels so you can easily integrate your solutions into a digital nervous system. At the hardware layer, we provide options for wireless microcontroller level connectivity with easy programming and updates. At the data aggregation and processing layer, our gateway appliances provide processing, data aggregation, and network bridging to ensure your solution can communicate regardless of protocol or medium. This layer is a critical difference for many applications, as the gateway appliance often acts as a connection point for web interfaces or enterprise software platforms. With a Synapse gateway you're not limited by protocol. If you need Bluetooth connectivity or just a TCP/IP stack, we've got you covered.

Building a connected solution is just the first part of the equation. A good design must be able to grow and evolve with user needs. That's why Synapse provides software tools for rapid prototyping and deployment as well as network management tools that help you maintain your connected designs once they're in the field.

This evaluation kit will provide you with an introduction to the **Synapse Network Application Platform**, or **SNAP**. **SNAP** provides a communications and processing platform for building IoT applications. It resides on **SNAP Engines** like the ones in this kit, which provide the connection point between your design and the physical world.

Thanks for checking out what Synapse and **SNAP** have to offer. We look forward to working with you!

Before Getting Started

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:

- a working familiarity with the included **SNAP** hardware
- installed the companion **Portal** software and any needed device drivers
- gained familiarity with the basics of using **Portal** and **SNAP** to develop wireless applications

Key Concept

The goal of this document is to get you to a point where you understand how a **SNAP** module can be incorporated into your development workflow and take advantage of the possibilities it brings.

Because this manual 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 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 are 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 **EK2100-220 Users Guide**, is only one of several that support this evaluation kit. (Most are installed on your system when you install **Portal**, which you will do as part of this tutorial.) Be sure and take a look at:

The “SNAP Primer”

This document contains an introduction to **SNAP** and explanations of how mesh networking works. It introduces the Synapse software and hardware, and clarifies naming conventions used for **SNAP** items.

The “SNAP Users Guide”

This document explains how the components in a **SNAP** network work together, with introductions to topics like programmatic response to sensed events, and sample scripts.

The “SNAP Reference Manual”

This document is essentially the **SNAP** API, and contains information on the built-in functions provided by **Portal** and **SNAPPy**. It also provides information specific to each platform **SNAP** has been ported to.

The “SNAP Sniffer Users Guide”

A “wireless sniffer” capability is included with **Portal**. It provides insight into the wireless exchanges that are taking place between **SNAP** nodes. It serves as a critical debug tool that allows you to “see” the network traffic flowing among the nodes in your **SNAP** network.

The “Portal Reference Manual”

This document contains lots of information on how to use **Portal**, our software for configuring and managing **SNAP** wireless mesh networks.

You may also wish to refer to the following hardware guides:

The “SNAP Hardware Technical Manual”

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

The “SN132 Quick Start Guide” and the “SN171 Quick Start Guide”

These two documents are subsets of the “**SNAP Hardware Technical Manual**” and come in handy because each focuses on a single board type.

The “RF220 Datasheet”

The specifics of the **RF220** hardware and working with it are outlined here.

All of these documents are in Portable Document Format (PDF) files and are available on the Synapse Wireless forum at forums.synapse-wireless.com.

Other Sources of Information

There is a dedicated support forum at 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.

A Note about the “Spirit” of this Development Kit

Our team has a well represented contingent of people who like to experiment and break things. We have a proud history of scaring our parents by taking apart televisions and washing machines. We’re the type of people who would REALLY like to take a lightsaber apart and see what’s inside. It’s how we learn, it’s how we roll.

The easiest way to get a solid understanding of **SNAP** and what it can do for you is to use the demonstrations in this document as a starting point for your own applications. We encourage you to tinker with the example code and see what happens. Change things up, add and subtract code, see what happens. Breaking the code is one of the easiest, most fun ways to learn.

Have a good time and remember, if you can break it, it probably needed to be fixed anyway.

Getting Started

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

SNAP is the **Synapse Network Application Platform**, our suite of tools for creating and maintaining Internet of Things (IoT) applications. 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 **SNAP Connect**.

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 be an integral “F” antenna, or a mounting point for an external antenna. The **SNAP Engines** included in your **EK2100-220** evaluation kit are **RF220-UF1 SNAP Engines**, (also known as **RF220 SNAP Engines**), as pictured above. **RF220 SNAP Engines** communicate using 802.15.4, 2.4 GHz radio signals.

If you have other Synapse platform hardware available, such as the **RF200 SNAP Engines**, you may be able to substitute them for the **RF220 SNAP Engines** in this kit. But it may be necessary to modify the sample code slightly to account for the platform differences.

Each **SNAP Engine** has an on-board microcontroller with its own internal memory. No external components are required for operation.

SNAP Engines include General Purpose I/O (GPIO) pins, which can be configured as digital inputs or outputs, and many GPIO pins can be switched to alternate functionality.

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

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

Although 20 I/O pins are available on the **RF220**, you only have to hook up the functionality required by your application. The minimal hookup to a **SNAP Engine** consists of two wires:

- One wire for VCC
- One wire for GND

SNAP Engines contain a core operating system(OS) that implements basic wireless networking functionality. This OS 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**” and “**SNAP Reference Manual**.” 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, responding to commands from your main microcontroller or microprocessor. In many cases the **SNAP Engine** can replace 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.

Two different kinds of **SNAP** demonstration boards are included in this evaluation kit:

- One **SN132 SNAPstick USB Module**
- 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 that runs on any standard PC with Microsoft Windows 7 or higher. Using a USB or RS232 interface, it connects to any **SNAP Engine** in a **SNAP** Wireless Network, becoming a user interface for the entire network.

Note: Most of the icons shown in these diagrams are taken directly from the **Portal** user interface.



is used (by default) within **Portal** to generically represent a **SNAP Engine**.



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

Once connected, Portal provides the capability to interactively build an intelligent sense-and-respond 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

Interactive – 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.

Sense – **SNAP Engines** contain an embedded microcontroller, allowing you to attach almost any sensor and interact with it wirelessly.



Respond – Scripts running on the **SNAP Engine** can provide warnings and even issue commands to other nodes to respond when a particular action is sensed. For example, a light can be switched on in response to motion, or water might be shut off in response to moisture being detected. Using **Portal**, you can even wirelessly change the programmed response of a node.

Synapse's **Portal** administrative software is freely downloadable from forums.synapse-wireless.com.

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.

Evaluation Kit Hardware

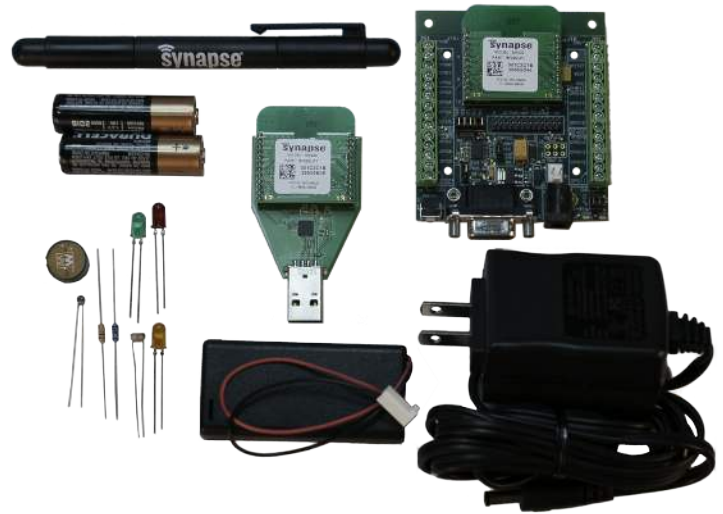
To demonstrate many of the capabilities of Synapse **SNAP Engines**, **SNAP**, and **Portal**, we've bundled them together in an evaluation kit form – the **EK2100-220**.

The **EK2100-220** evaluation kit contains two different kinds of **SNAP** demonstration boards, each with an **RF220 SNAP Engine**:

- One **SN132 SNAPstick USB module**
- One **SN171 Proto Board**

The **EK2100-220** evaluation kit also comes with:

- a power supply
- an external battery holder
- a pair of AA batteries
- a screwdriver



Remember: You do not have to use the various “demo” boards in order to use the **SNAP Engines** in your own projects.

Also: If you crave even more fun, 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 components bag contains 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 (little ball on two sticks)
- 1 Piezo buzzer (small black disc)



SN171 Proto Board

The Synapse **SN171 proto** (prototyping) **board** provides easy (terminal block) access to 19 of the 20 General Purpose Input/Output (GPIO) pins available on the **RF220 SNAP Engine**. All 20 GPIO pins are accessible via the dual-row header in the center of the board. Seven of these pins (GPIO 11-13 and 15-18) can also serve as analog inputs.

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.



SN132 SNAPstick USB Module

The Synapse **SN132 SNAPstick USB module** (often also called a Paddleboard because...well...look at it...) does not provide access to the entire range of GPIO pins available through the **SN171 Proto Board**. Instead, it provides a simple, compact way to connect a PC running **Portal** to a **SNAP** network via USB.

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 (GPIO_0)	Value of GPIO Pin 1 (GPIO_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 doesn't need to be installed for the SNAPstick to draw USB power.

More information about this module's hardware can be found in the **SN132 SNAPstick Quick Start Guide** and the **SNAP Hardware Technical Manual** (available from the Synapse website).

Demonstration Time

The following section takes you 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.

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

SN132 USB SNAPstick Connection

Connect the **SNAPstick USB module** to an open USB slot on a 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.

For Windows Vista and later versions, the drivers should automatically be installed the first time the SNAPstick is plugged in. If not, the drivers will get installed when **Portal** is installed, which is discussed later in this document. If you're using something older than Windows Vista and get the dialog box, just click **Cancel** for now.

Demonstration 1: Counting Straight Out of the Box

The nodes on the **SN132 SNAPstick** and **SN171 Proto Board** are pre-loaded with the *McastCounter.py* demo **SNAPpy** script. This demo script showcases a small portion of the **SNAP** node's capabilities and gives you a chance to get familiar with the nodes without installing any software on your PC.

The Components: <ul style="list-style-type: none">• RF220 Modules• SN171 Proto Board• SN132 SNAPstick	What we'll do: Just plug some stuff up. Easy!
	Key Takeaway: SNAP nodes interact without connecting to a computer or any other network. They work straight out of the box.

Instructions

Note: Don't install the **Portal** software just yet. If you have this module plugged into your PC, it will draw power from the USB connection. However it is not yet interacting with any installed software. **SNAP Engines** do not require **Portal** to operate. Each **SNAP Engine** is an autonomous (and customizable, through scripting) node in a network.

Step 1 – Plug the **SN132 SNAPstick USB module** into a PC or USB power supply. A single green power indicator LED on the module will light to indicate it's powered.

Keep the **SNAPstick** 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.

Step 2 – Power up the **SN171 Proto Board** using the AC adapter provided with your evaluation kit.

A yellow LED on the **SN171** (LED2) will begin to flash.

Step 3 - The *McastCounter.py* script keeps track of a global count. This count is incremented every time a 'button press' is registered on **any** button-equipped module in the mesh and running the *McastCounter.py* script.

Push the button on the **SN171 Proto Board**. The LEDs on the **SN171 Proto Board** and the LED on the **SN132 SNAPstick** 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 LED blinks in the same pattern.

If you power cycle the **SN171 Proto Board**, you can see that the LED on the **SN171 Proto Board** begins to flash again waiting for you to press the button to change the pattern.

Key Concept

A **SNAP** module can stand alone as a microcontroller. Unlike many microcontrollers, it's easy to communicate with because of the **SNAP** platform.

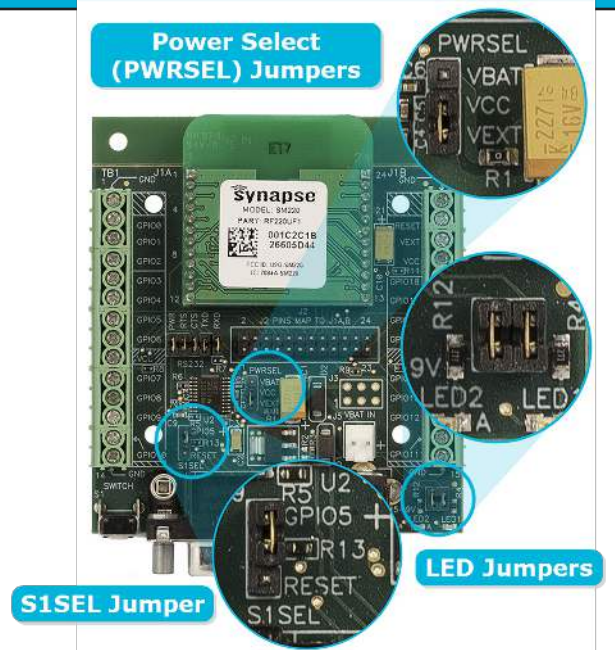
So...what just happened?

The McastCounter.py script on the RF220 modules on the **SN132 SNAPstick** and **SN171 Proto Board** starts as soon as the modules get power, and the modules can immediately talk to each other wirelessly thanks to the **SNAP** operating system. They don't need to hear from a computer or other component to start up.

Each module is listening for a signal that a button is pressed, and each responds in a pre-programmed fashion. Note that this happens on the **SN132 SNAPstick** despite it not having a button. It still recognizes the command from a connected node.

Wait! It didn't work

Problem	Action	Description
The LEDs didn't illuminate	Verify the power related jumpers	Your SN171 Proto Board 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).
	Verify the LED related jumpers	The SN171 Proto Board 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 both LED jumpers are installed.
The LEDs are lit, but they're not changing	Make sure the button is enabled	If you are not seeing the LEDs change on the SN171 Proto Board make sure that the button is enabled. The S1SEL jumper located up and to the right of the button should be set to 'GPIO5' (connecting pins 1 and 2) and not to 'RESET' (pins 2 and 3).
I can't stop pushing the buttons and watching the LEDs change	It's likely that you appreciate awesome technology	This isn't a bad thing. In fact, you'd fit right in with the crew here at Synapse. Take a look at www.synapse-wireless.com/careers/ and see if we have an opening for your field of expertise.



Summary and Next Steps

This demonstration is a simple way to show some 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 software is required. There is no need for a PC to “coordinate” the nodes; they 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 also 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.

Installing Portal

This kit includes 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 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 www.synapse-wireless.com

Running Setup

Unplug any Synapse devices from your PC, then download and run the **Portal** installer.

Depending on the version of Windows running on your PC, you may get a warning dialog asking if you wish to run the file. The warning is harmless. Click **Yes** to proceed with the installation.

Note: You likely know how to install software, and **Portal** installation is pretty routine. However, we do have one warning in the text below that you'll miss if just hammer straight through.

WARNING: You'll need to install Bridge drivers before running **Portal** for the first time. More information is at the end of this section.

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.

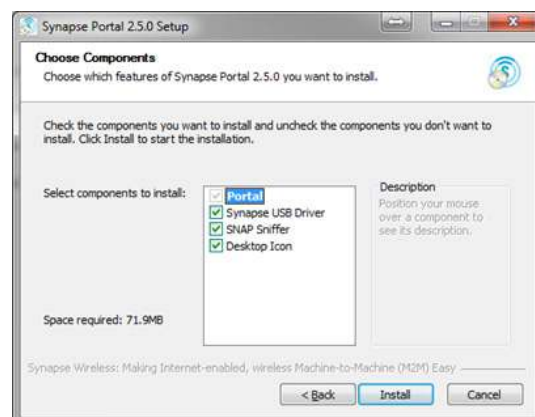
When the intro screen appears, click the **Next** button to load the license agreement pane. Read the license agreement at the specified URL, check the **I agree** box, and then click **Next**.

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

You'll now have the option of also installing the Synapse USB Driver and the **SNAP Sniffer**. They're both pretty handy, so we suggest that you go ahead and install them. The sniffer is actually necessary for later exercises in this document so consider it a time saver.

As for the Desktop Icon, that's up to you and whether or not you want stuff on your desktop. There's no right answer. We won't judge.

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 a dialog box asking if you'd like to install them. Considering nothing works right without them, you should probably go for it. It'll make the rest of this tutorial WAY easier.

To ensure that the latest Synapse USB drivers can be installed, you can't be running the old versions of these drivers. Disconnect any Synapse USB devices (the **SN132 SNAPstick** from exercise one is probably still plugged in) to ensure this, and then click the **Install** button. The installation process will continue.

If everything works the way it should, you'll get a message saying that **Portal** is installed. There will be a **Portal** icon on your Windows Desktop (if you specified there should be), as well as in the Start Menu. Don't run **Portal** just yet though. You need to complete the bridge device driver installation. Uncheck the **Run Synapse Portal** checkbox before clicking **Finish**.

Plug in the Bridge Device


The **SN132 SNAPstick** will serve as a bridge between your PC and the **SNAP** network.



Plug the **SN132 SNAPstick** into a USB port on the PC that will run **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 the **SN132 SNAPstick** are automatically downloaded the first time it is plugged in.

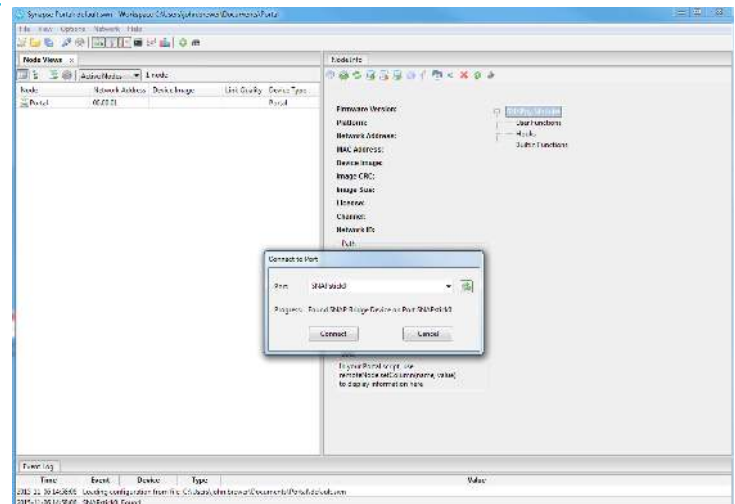
Note: If the drivers for the **SN132 SNAPstick** are not automatically installed when plugged in for the first time, please download and install the latest FTDI driver from the website www.ftdichip.com/Drivers/VCP.htm

Launching Portal

After installation, launch the **Portal** program. You should see a screen similar to the one at right.

If you do, you have successfully installed **Portal** and detected a bridge device connected to a USB 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. Still with us? Good.



Now that **Portal** is installed, let's talk about how to use it!

Key Concept

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** connects serially to this bridge node, using either a USB or RS232 connection.

Portal is then able to communicate to the rest of the **SNAP** nodes indirectly by sending packets across the directly connected bridge node. For our purposes, the **SN132 SNAPstick** acts as the bridge.

Using Portal

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

We'll continue using the **SN132 SNAPstick** from the previous section, so keep it connected to the USB port.

Press **Connect** once the bridge device (**SN132 SNAPstick**) is detected (this is probably going to be USB0).

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 display choices for **SNAP Reference Manual**, **Portal Reference Manual**, etc.

The convention is that menu choices ending in “...” usually display 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 where 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

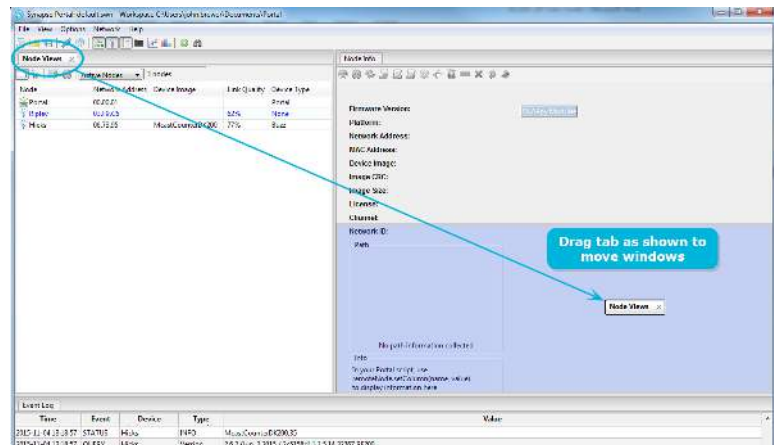
The remainder of the **Portal** GUI is taken up by a changeable collection of tabbed windows. Each of these windows has a name, which is displayed in the tab for that window.

Many of the tabbed windows have tool bars 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

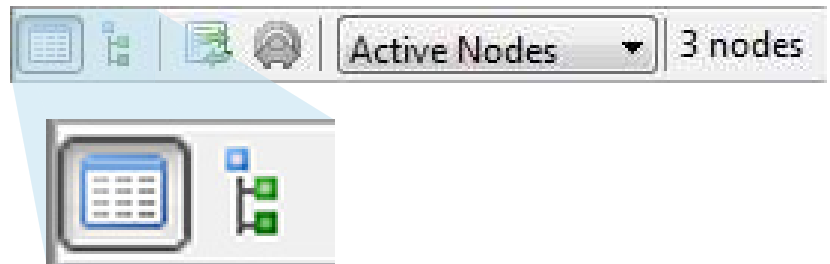
You can close tabbed windows that you no longer want by clicking the small **X** located to the right of the name in the tab when the tab has focus.

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 two buttons for now.



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

-  **Report View**
-  **Tree View**

Both 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  button in the main tool bar to refresh it.

Click the  **Report View** button.

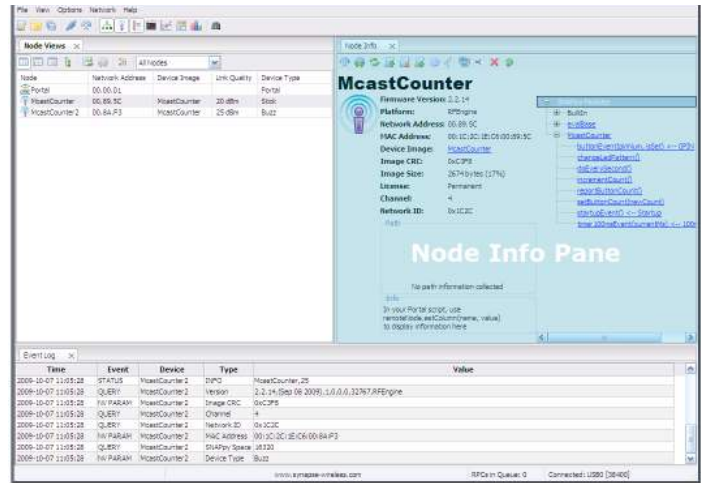
You should see that two devices have been discovered. They all will have names of the form McastCounter. The directly connected “bridge” device (the **SN132 SNAPstick**) should be shown in blue, while the remote device is displayed in black. The nodes report in using a random response delay, so which node is named McastCounter and which is named McastCounter2 can vary.

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 “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. 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.

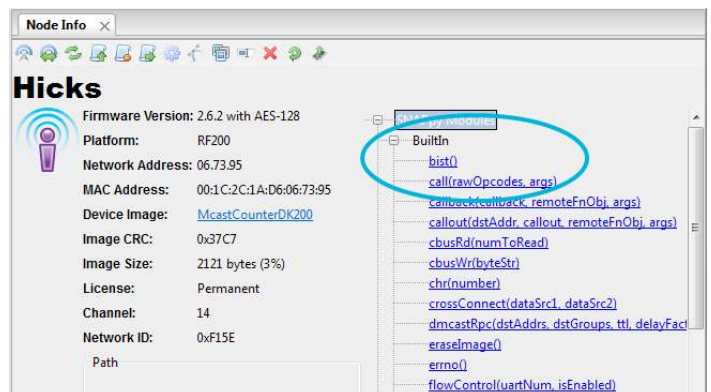
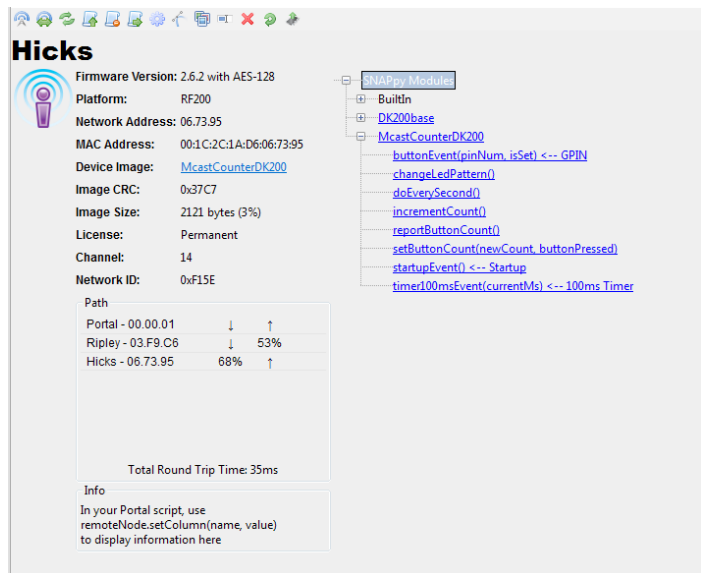
On the left-hand side, the **Firmware Version, Platform, Network Address, MAC Address, Device Image, Image CRC, Image Size, License, Channel, and Network Id** are shown. Below that is a block where **Path** information (the path to/from the node) can be displayed. Below that there is an **Info** field 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” is loaded into the node. You can click the device image name shown (McastCounterDK200), and automatically load that script in **Portal**'s built-in source code editor.

On the right hand side, a collapsible tree of available functions is shown. In the screenshot at right, 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**.



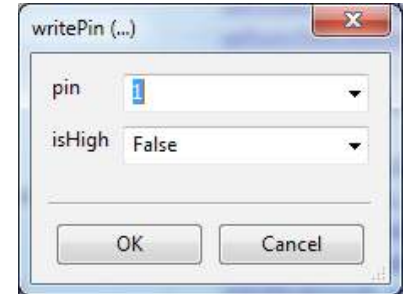
Functions that do not require any parameters (for example, the `getChannel()` function) will be executed immediately.

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” and “isHigh = True”) and press **OK**, or
2. You can press **Cancel** to abort the function invocation.



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

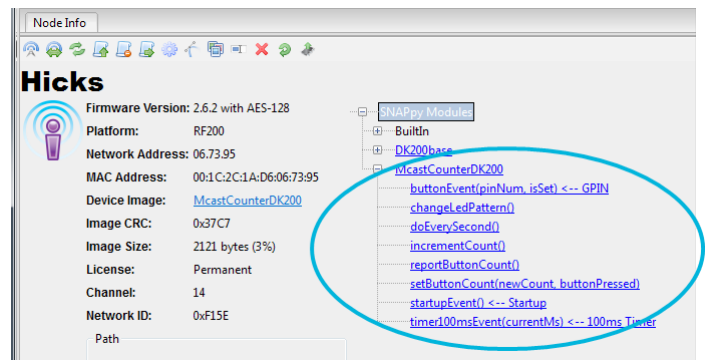
You can also expand the tree of functions defined by each module (in other words, by each **SNAPpy** source file).

Text fields in **Portal** maintain a history of entered values to facilitate making a particular function call or node configuration again. When you invoke a text field, **Portal** will show you the most recent previous value used. If more than one value has been specified in the field since **Portal** was installed or since the GUI component history was cleared, **Portal** will display a down arrow indicating that there are other options available for selection.

Here you can see the various functions defined in the “McastCounterDK200” **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 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 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 **SN171 Proto Board** node in the **Node Views** panel. Then, in the **Node Info** panel, select “Upload **SNAPpy** Image” .

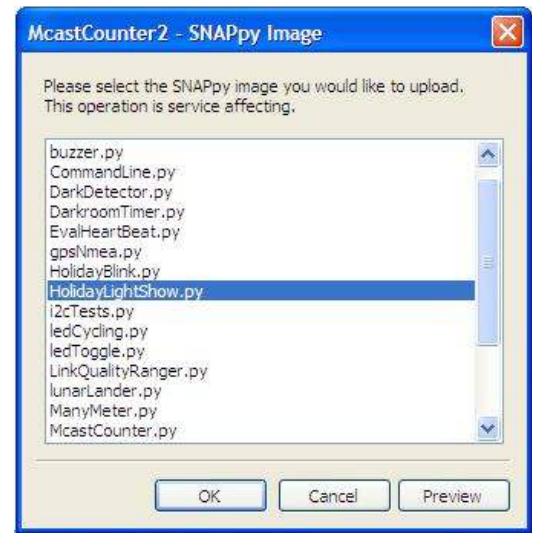
(If the button is disabled, it means **Portal** does not know enough about the node to enable it to load a script into it. Click the **Refresh Node Information**  button, and after the node information refreshes the **Upload SNAPpy Image** button should be enabled.)

This will display a dialog box asking which script/image to upload. These scripts are located in a `Portal\snappyImages` directory within your “My Documents” folder.

Select the “HolidayLightShow.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 “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, 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

Let’s take a look at how sensors allow nodes to interact with **Portal** and each other.

You should now have the **SN132 SNAPstick** running the example script “HolidayBlink,” and the **SN171 Proto Board** running another example script, “HolidayLightShow”. If you do not, please refer to the previous section **Uploading SNAPpy Images**. (...and quit skipping steps!)

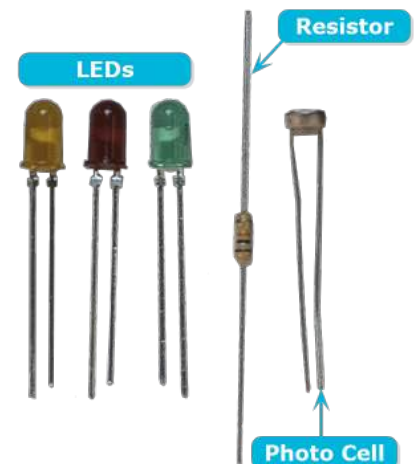
Until now we have ignored the **SNAPpy** (pun intended) little Synapse screwdriver and small bag of electronic components that accompanied the kit. Now is the time to roll up our sleeves and get to work on a second demonstration.

<p>The Components:</p> <ul style="list-style-type: none"> • The stuff from Demonstration 1 • 1 Red, 1 Green, and 1 Amber LED • 1 10K Ohm resistor (the beige one) • 1 Photo cell 	<p>What we’ll do:</p> <p>Demonstrate how information gathered from sensors onboard a SNAP node can be communicated to other nodes and used to initiate other tasks</p> <hr/> <p>Key Takeaway:</p> <p>SNAP creates a sense-and-respond network, which allows nodes to react based on sensor data and instructions from other nodes.</p>
---	--

Instructions

First, make sure the screws in the **SN171 Proto Board** 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. (As popularized by the ever-popular statement “righty-tighty, lefty-loosey.” If you didn’t know it before, it’s forever stuck in your head now.)

We’ll use the pins on the left hand side of the **SN171 Proto Board** to connect the LEDs. We’ll also use the pins in the lower right hand corner of the **SN171 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 that indicate how much resistance they provide. 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.

Step 1 – Disconnect power from the **SN171 Proto Board**. (Leave the **SN132** connected.)

Step 2 – Connect the Green LED

Find the green LED. LEDs have a polarity and can only be connected in one way, so we'll need to determine which of the two leads is the negative terminal (a.k.a the cathode). One of the leads of the LED is shorter than the other. This is the “negative” leg. A flat spot on the rim at the base of the colored bulb also indicates the negative side.

Place the shorter “negative” leg of the green LED into the terminal block at the connector labeled GPIO 8 (the fourth connector up from the bottom of the left-hand side of the **SN171 Proto Board**).

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

Tighten the screws located on the top of the terminal block for both GPIO pins. This will secure the LED to the **SN171 Proto Board**.

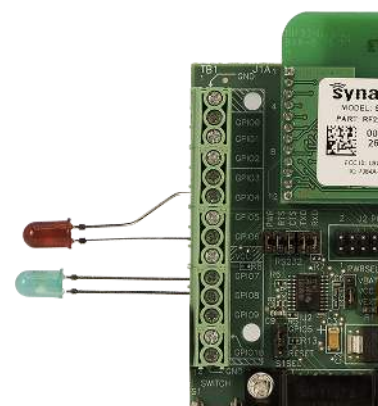


Step 3 – Connect the red LED

Place the “negative” leg of the red LED into the terminal block at the spot labeled GPIO 6 (near the middle of the left-hand side of the **SN171 Proto Board** — note that it is NOT next to GPIO 7; there is a VCC connection between them).

Place the other leg of the red LED into the terminal block at the position 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.

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



Step 4 – Connect the amber LED

We know you've got this one, but just in case...

Place the “negative” leg of the amber LED into the terminal block at the position labeled GPIO 3.

Place the other leg of the amber LED into the terminal block at the spot 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.

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

Step 5 – We are now ready to interact with the **Portal** software we installed in the previous section of this document. (If you haven't installed **Portal**, please go back now, and quit skipping steps!)

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 to the **SN171 Proto Board**, then switch to the **Node View** pane in **Portal** that was described in the previous section.

Select the **SN171 Proto Board**, which has its 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 the list of available functions. (You might need to collapse the “evalBase” section of functions, or scroll down to find the right functions.)

Click the **startChristmasDisplay()** function.

The green and red LEDs will begin to blink.

Merry Christmas!

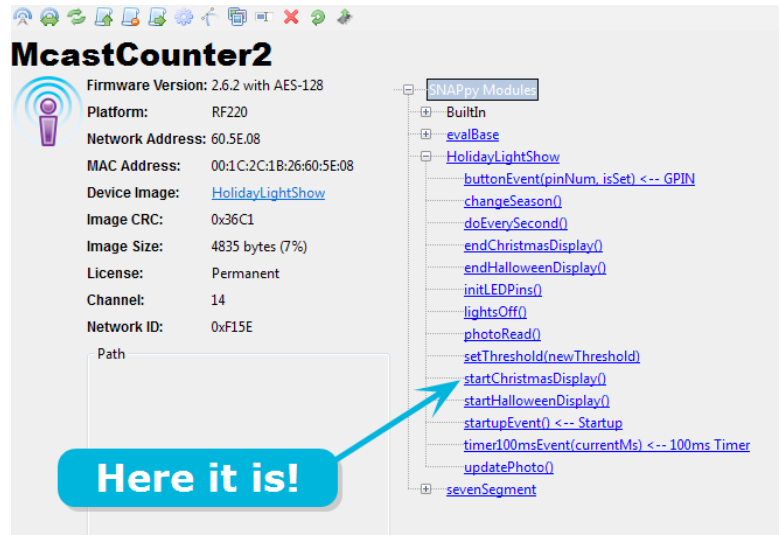
Click the **changeSeason()** function.

The amber and red LEDs will 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 Proto Board**'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.



Note: We are aware that Halloween comes before Christmas, but we like Christmas better, mainly because of the presents and lack of “tricks.” If you’d like to extend the exercise by switching the order that they appear by all means mod the functions to your liking. We won’t mind, but we don’t get into the code until a later exercise. If you’ve been wanting to “jump ahead,” now is good. The management of Synapse Wireless has no official position on the matter of Christmas over Halloween, and just looks at us funny when we ask.

So...what just happened?

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?

Ok...we think it’s pretty cool. If you think about the normal hassles of wireless communications between devices it’s VERY cool.

For the time being we’ll need to disconnect the power to the **SN171 Proto Board**.

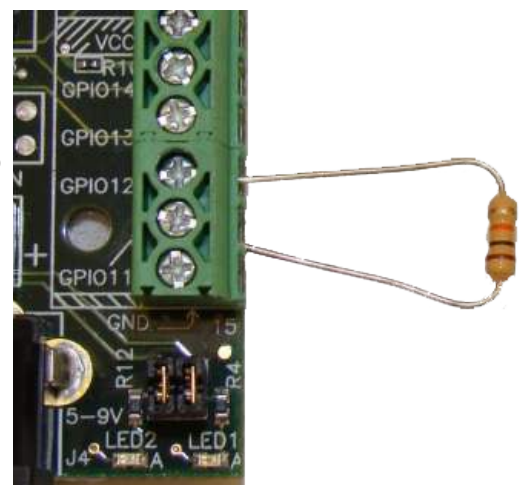
Step 6 – Connect the resistor

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

Place one resistor leg into the terminal block at the pin labeled GPIO 12 (the third spot up from the bottom of the right-hand side of the **SN171 Proto Board**), **but don’t tighten the screw yet**.

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

Now, tighten the screw for GPIO connector 12 only. This should lock the one leg of the resistor in place. We’ll address the other leg in a moment.

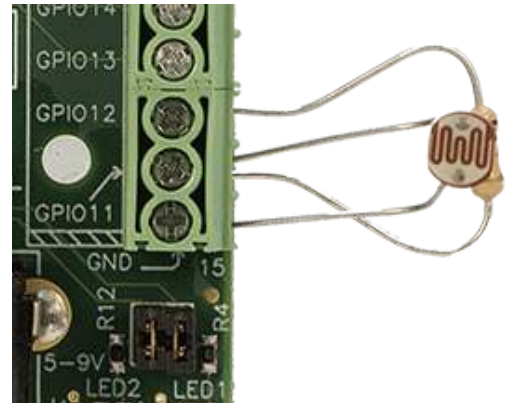


Step 7 – Connect the photo-cell

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

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.



Historical Note: That phrase was coined in 1911. When adjusted for inflation, a picture is now worth about 41 words. We expect that cell phones have flooded the photo market. Still, pictures are pretty handy.

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 8 – Reconnect power to the SN171 Proto Board.

Step 9 – The “HolidayLightShow” script is already monitoring for changes in light levels on the **SN171 Proto Board** (sneaky, we 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 light/darkness.

Step 10 – 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 **SN132 SNAPstick**. You will notice that the green and red LEDs are lit up as well. The **SN171 Proto Board** has communicated the change in sensor status over the air to the neighboring device to extend the Christmas show.

Note: Sometimes holding your finger close to the sensor will trigger it twice. If you see the amber 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, meaning 85%.)

Step 11 – Change seasons

Move your hand away and then back to hovering over the sensor again. A Halloween light show (amber and red LEDs) will begin to flash. As before, the **SN171 Proto Board** has wirelessly communicated the message to change the light pattern on the **SN132 SNAPstick**.

Finally, the button on the **SN171 Proto Board** turns the lights off, or you can execute the `lightsOff()` function 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 Concept

Each module is part of the network, and can issue instructions even when it isn't configured the same as the receiving node. We'll explore this more in Demo 4.

Note: You are welcome to use this exercise to run a larger holiday light show, but the nightly switching between Halloween and Christmas will probably be confusing to your neighbors.

This, combined with the daylight savings time shift that occurs between the two holidays, may annoy your neighbors to no end. If this is your goal, by all means continue. Just so you know, Synapse is not responsible for what your neighbors think, or do, in response.

Wait! It didn't work

Problem	Action	Description
One of the LEDs isn't working	Verify the LEDs are installed correctly	LEDs have a positive and a negative lead. (The negative lead is the short one.) Make sure that the LEDs are in their proper numbered slots on the terminal block, and the negative and positive pins are in the correct slots.
The LEDs aren't working, and neither are the lights on the SN171 Proto Board	The board isn't powered	Make sure it's plugged into a power supply, (wall or battery) and the jumpers are set correctly for that power type.
I'm pretty sure the lights are saying something in Morse code	You've been trying to find a working IoT solution for a really long time	We understand. Keep reading. We'll get you up and running with SNAP and your stress level will probably go down. We hope so at least.

Key Points

- **SNAP Engines** can read and respond to analog inputs.
- 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.
- Nodes can trigger events on other nodes, even though they're not configured the same.

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 message to a group of devices.

This particular example uses a multicast Remote Procedure Call (RPC). A uni-cast RPC (instruction sent to only one node) 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** and **SNAP Users Guide**.

Demonstration 3: Temperature Alarm

One of the key strengths of a wireless mesh network is the ability to share information and components among nodes. In this exercise, you'll see how a node with a temperature sensor can trigger a buzzer when a parameter is exceeded.

The Components: <ul style="list-style-type: none">• Everything from Demonstration 2• Buzzer• Thermistor• 100K Ohm Resistor	What we'll do: <p>Add some components to our current setup and make it into something that can be hacked into whatever you want.</p>
	Key Takeaway: <p>Nodes become more powerful when they're programmed to work together.</p>

The **SN171 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.

Instructions

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

We'll use the connections in the upper right hand corner of the **SN171 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 add the blue resistor for this demonstration.

Step 1 – Disconnect the power running to the **SN171 Proto Board**.

Step 2 – Connect the buzzer to the **SN171 Proto Board**:

Place the leg of the buzzer (see picture) marked with a '+' into the connector at GPIO 9 and the other leg into the connector marked GND.

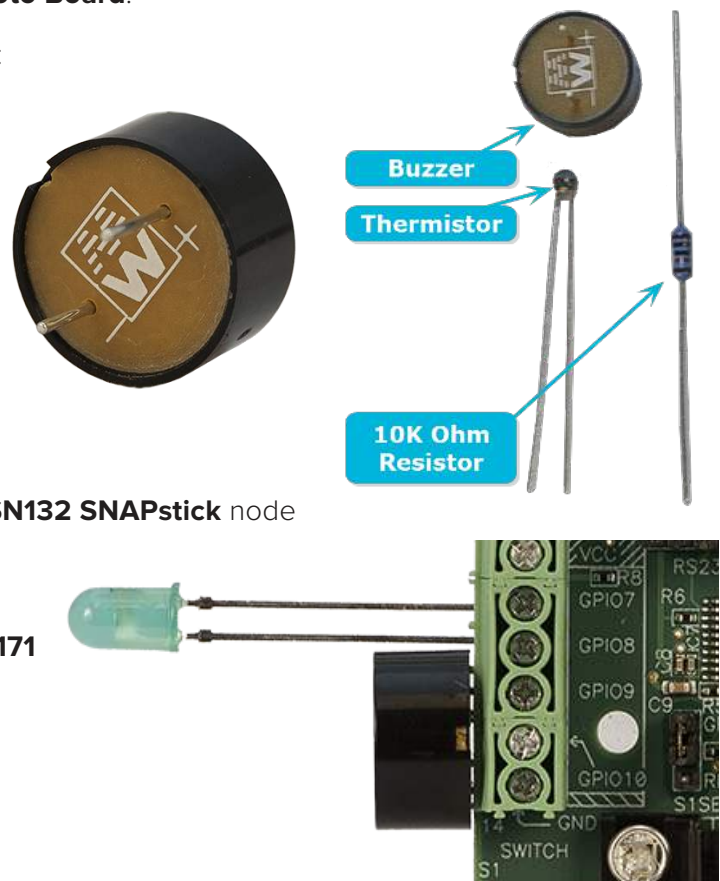
Tighten the associated screws and try a gentle tug on the buzzer. It should not move, but be firmly connected to the **SN171 Proto Board**.

Step 3 – Use **Portal** to upload new scripts.

Connect the power back to the **SN171 Proto Board**.

Load the script "TemperatureAlarmBridge.py" onto the **SN132 SNAPstick** node using the step you learned in the **Uploading SNAPpy scripts** section.

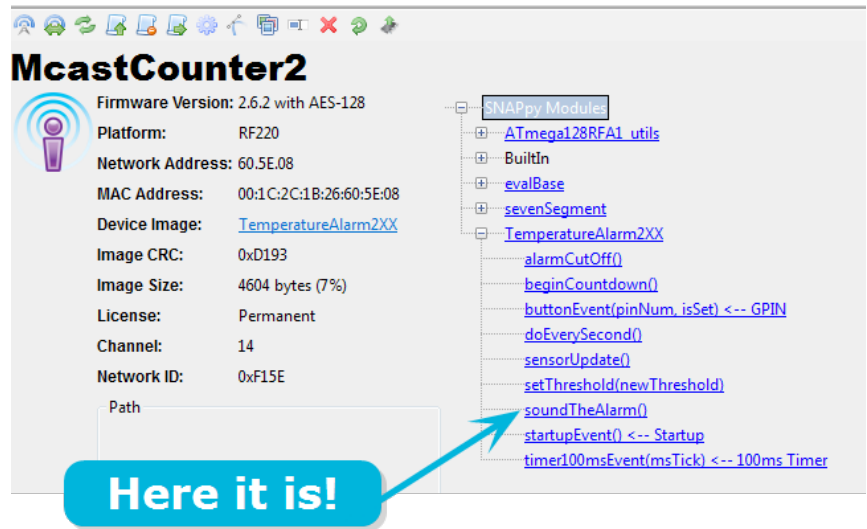
Load the script "TemperatureAlarm2XX.py" onto the **SN171 Proto Board** node.



Step 4 – Make sure that you still have the **SN171 Proto Board**'s node (the one with “TemperatureAlarm2XX” in it) selected in the **Node View** pane.

Switch to the **Node Info** pane and find the “TemperatureAlarm2XX” 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 Proto Board** again.

Step 5 – Connect the resistor (blue) to the SN171 Proto Board

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

Place one resistor leg into the terminal block labeled GPIO 18 (located on the upper right-hand side of the **SN171 Proto Board**.)

Place the other leg into the connector labeled VCC (located in the center of the right-hand side). There is no polarity on a 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 between GPIO 14 and GPIO 15.

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?)

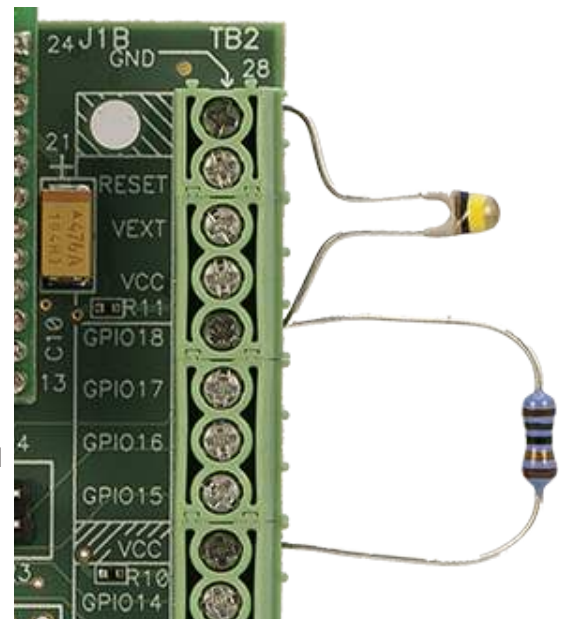
Step 6 – Connect the thermistor to the SN171 Proto Board

Place one leg of the thermistor (see picture for identification) into the spot labeled GPIO 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 some number of 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 – Reconnect the power 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 Proto Board** and the flashing amber LED on the **SN132 SNAPstick**.



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

1. Executing the `alarmCutOff()` function in the “TemperatureAlarm2XX” specific section of the **Node Info** pane within **Portal** (this can be done on either node).
2. By pressing the built-in **SN171 Proto Board** button.

A quick ‘chirp’ of the buzzer will indicate that the alarm is bypassed (much like a car alarm).

Wait! It didn't work

Problem	Action	Solution
The buzzer isn't working	The polarity on the pins is reversed	The buzzer has a positive and a negative lead. (Marked as a + and a - respectively on the back of the buzzer.) Make sure that the pins are in their proper numbered slots on the terminal block, and the negative and positive pins are in the correct slots.
The buzzer is really annoying	Your hearing is probably within a normal range.	It annoyed us too. Unfortunately an Internet of Things solution requiring an audible alarm really requires a...well...audible alarm. Try putting some tape over the hole, or maybe you could stash the SN171 Proto Board in a coworker's office and randomly trigger it just to pester? We never get tired of that one.
The Modules have achieved a level of sentience and they're trying to eradicate humanity	You have some SERIOUS Python skills!	Many of us have wondered if we're enabling some sort of “rise of the machines” scenario, but you've actually done it! Bravo! Try and remind our new overlords that you were instrumental in their rise. Hopefully that will be worth something. Also, remember the classic “hydraulic press” and “sink-em in molten metal” methods. Those are time-tested and proven defenses.

Key Points

- Synapse **SNAP Engines** can read and respond to multiple 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.

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 **SN171 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.*


The Components: <ul style="list-style-type: none">• Everything from Demonstration 3 (See? Following the steps in order was a good idea!)	What we'll do: Use the SN171 Proto Board we've already constructed to do something entirely different.
	Key Takeaway: It's easy to reprogram a SNAPpy node, and they can communicate information 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 **SN171 Proto Board**.


Upload the “ManyMeter2XX.py” **SNAPpy** script onto the **SN171 Proto Board** (a skill we've mastered from previous sections, but here is a hint: ).

Let's give the **SN171 Proto Board** 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 **SN132 SNAPstick**.

Switch to the **Node Info** pane and erase the script currently running on the **SN132 SNAPstick** 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 **SN132 SNAPstick** 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 Proto Board**) 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.

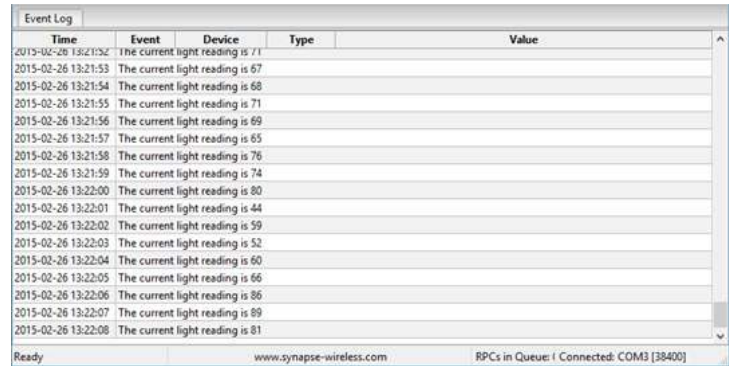
Calibrate the sensor once again by covering it to simulate “complete” darkness. Uncover the sensor.

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

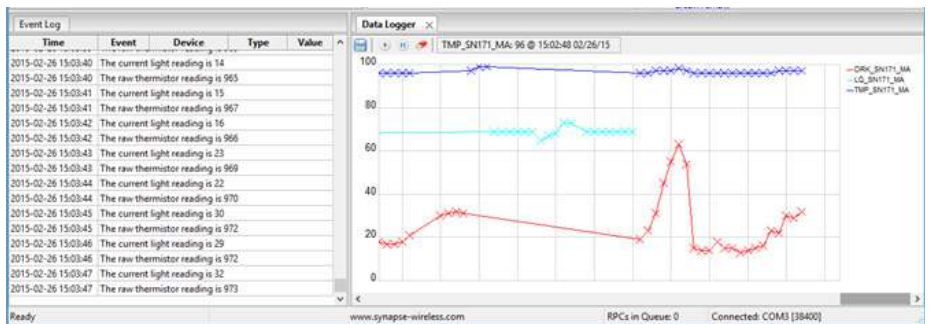
Push it again and the radio link quality will be displayed in dB.

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.



Time	Event	Device	Type	Value
2015-02-26 13:21:52	The current light reading is	/ 1		
2015-02-26 13:21:53	The current light reading is	67		
2015-02-26 13:21:54	The current light reading is	68		
2015-02-26 13:21:55	The current light reading is	71		
2015-02-26 13:21:56	The current light reading is	69		
2015-02-26 13:21:57	The current light reading is	65		
2015-02-26 13:21:58	The current light reading is	76		
2015-02-26 13:21:59	The current light reading is	74		
2015-02-26 13:22:00	The current light reading is	80		
2015-02-26 13:22:01	The current light reading is	44		
2015-02-26 13:22:02	The current light reading is	59		
2015-02-26 13:22:03	The current light reading is	52		
2015-02-26 13:22:04	The current light reading is	60		
2015-02-26 13:22:05	The current light reading is	66		
2015-02-26 13:22:06	The current light reading is	86		
2015-02-26 13:22:07	The current light reading is	89		
2015-02-26 13:22:08	The current light reading is	81		



Press the button on the **SN171 Proto Board** to cycle through each available sensor reading and watch **Portal** graph the information. Notice that multiple readings can be displayed concurrently. Every fourth button press will disable the display of information.

Helpful Reminder

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.

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 “ManyMeter2XX.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.

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. Just because you **CAN** do something, doesn't always mean that you **SHOULD**. The best way to build a GUI application is to use **SNAP Connect** rather than **Portal**. So, proceed with caution....

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 is already running 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.

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

Demonstration 4b: The Many-Meter extended

*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.*

The Components: <ul style="list-style-type: none">• Everything from Demonstration 4	What we'll do: Output data from the SNAP nodes into a display on your PC screen. It's like big data in action!
	Key Takeaway: A PC running Portal is also a node in the network, and can be leveraged for display options as well as extra computing power.

Step 1 – Switch back to the **Node View** pane in **Portal** and click the **Portal** node. Then click **Change Portal Base File**  in the **Node Info** window for the **Portal** node. Select the script called “PortalManyMeter.py,” located in the “My Documents\Portal” folder.. Then click **Open**. This action uploads the new file to the **Portal** node.


Note: **Portal** scripts are not, by default, in the same directory as **SNAPPy** scripts. (They're in the directory above the snappyImages directory where SNAPPy scripts are stored.) **Portal** is capable of loading nearly any Python (2.6) 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.

Step 2 – From the same **Node View**, verify that the **SN171 Proto Board** (node with the Device Name set to “OurDemo”) is still running the “ManyMeter2XX” script. Click the node.

Switch to the **Node Info** pane and click the script name next to the Device Image heading. This will load 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.)

1 Some standard libraries typically included with Python are not included in Portal.

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 is commented out (Python comments start with the `#` character). We have 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))
```

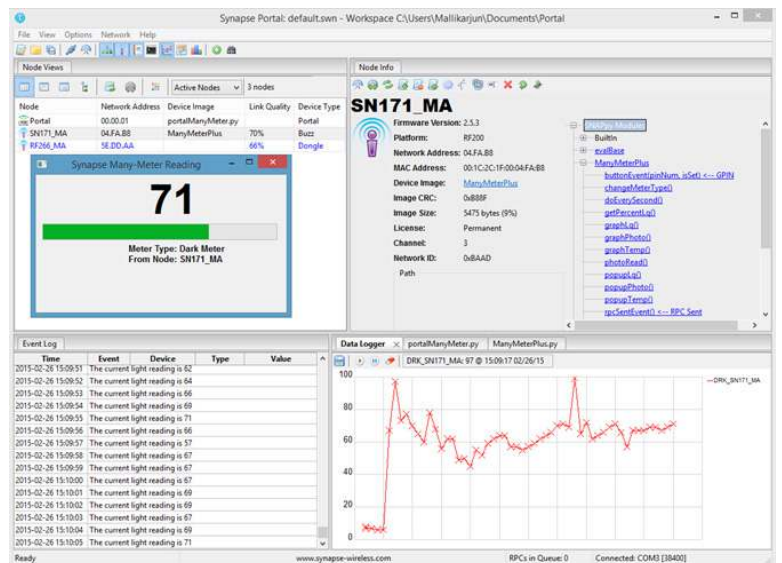
Repeat this process for the other 2 lines (for a total of 3 lines) with the same comment.

Save your work by clicking the **Save** icon  in the toolbar.

Step 4 – Upload your brand new script, “ManyMeterPlus.py”, to the **SN171 Proto Board** node (device name set to “OurDemo”).

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 could include:
 - Email
 - Output to database or log files
 - Graphical output to the screen

Alternative Energy Settings

Battery Operation

So far, we have been running the nodes from USB power (the **SN132 SNAPstick**) and wall power (the **SN171 Proto Board**.)

It is also possible to run the **SN171 Proto Board** from battery power using the battery pack included in the **EK2100-220** 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 **SN171 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

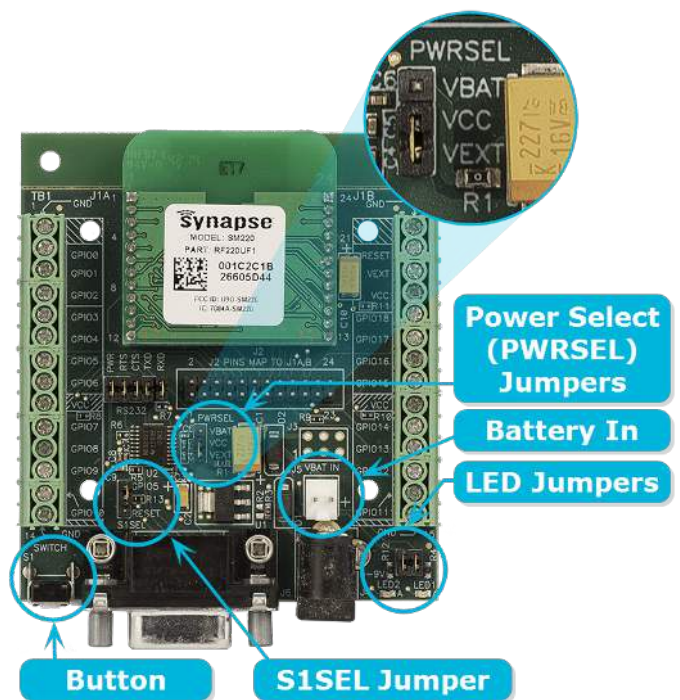
A **SNAP Engine** on the **SN171 Proto Board** is capable of achieving years of battery life from the included AA pack, but this requires running a **SNAPpy** script that sleeps, and removing all RS232 jumpers (JMP2, JMP5, JMP6, JMP7 and JMP8) from the **SN171 Proto Board**.

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.

Note: It’s important to remember that a sleeping node cannot respond to messages from other nodes, so you cannot use **Portal** to invoke a function on a node while it is sleeping. A button-push is a physical action and CAN wake a node from a sleep state.



Where To Go Next

In this manual we introduced the components of the **EK2100-220 Evaluation Kit**, installed **Portal**, and ran some simple demos.

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

- The “**SNAP Primer**”
- The “**SNAP Users Guide**”
- The “**SNAP Reference Manual**”
- The “**Portal Reference Manual**”
- The “**SNAP Hardware Technical Manual**”
- The “**SN171 Proto Board Quick Start Guide**”
- The “**SN132 SNAPstick 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?

Synapse also offers a larger evaluation kit called the **EK5100-220**. It includes a Synapse **E20 Gateway** and another **SN171** board for 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 **E20 Gateway** is particularly important if you'd like to explore how a **SNAP** enabled device can interact with the Internet and devices connected to the Internet. The **E20 Gateway** provides a full Linux computer for processing and memory intensive applications that go beyond what a **SNAP Module** can handle.

You can purchase these boards individually (outside of kit form) as additional nodes. They will interact with any of the included **EK2100-220** demonstration boards.

The **EK2100-220** kit only includes a single type of Synapse **SNAP Engines**, the F-antenna **RF220UF1**. Other forms of **SNAP Engines** that were mentioned in the introduction of this document can also be purchased on an individual basis.

Remember: Any of the **SNAP Engines** can be used interchangeably with any of the demonstration boards.

Visit us online

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

www.synapse-wireless.com

You can find an ever-expanding collection of useful information on the Synapse Support Forum at 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.