Quick Start Guide for E20 Example 1 – Gateway-Hosted Web Server

This example showcases the following products (for example, from a Synapse EK5100 kit):

- SNAP Connect E20
- SN171 Prototyping board, with RF200 module
- SN173 Prototyping board with SM220 module
- SS200 USB SNAP Stick
- SN132 USB SNAP Stick (loaded with SNAP Sniffer image)

The EK5100 kit ships with this example preloaded, or you can load it manually (*see page 2 of this Quick Start*).

What This Example Does

SNAPpy scripts running on the individual SNAP Nodes report button state, "button presses" (*counters*), and battery voltage. Python code running on the E20 Gateway implements a web server, and web browsers which connect to this web server display a table of live node data. Through that web page, you can control an LED on each SNAP Node.

Full source code for this example is available on Github here: https://github.com/synapse-wireless/demo-kits

The Synapse Portal IDE will allow complete embedded module development, as well as wireless sniffer capability – download the latest version here: https://forums.synapse-wireless.com/showthread.php?t=9

Running This Example

Simply power up the E20 and use a PC or mobile device to connect to its' WiFi access point:

SSID: synapse-e20
Password: synapse1

Open a web browser on that PC or mobile device, and point to the E20's URL: http://192.168.0.1

The web page will display a simple table of wireless SNAP nodes reporting their "status". The SN171 and SN173 boards run SNAPpy scripts which report status every 5 seconds, or when a button is pressed.

Connect the battery packs to the SN171 and SN173 boards, and verify each pack's switch is ON.

You should see a blinking LED on each prototyping board. Also, you'll see both devices show up in the HTML table displayed in your web browser.

As you press button-1 on each board, you'll see the press-count immediately updated in your browser. Also, the current state of each button will be reflected in real-time. In addition, the boards report their current battery level.

There is also a checkbox on each table row that controls an LED on the corresponding SNAP Node. Click it!

Loading This Example

Depending on its' model number, your kit may have come with this example *already preloaded* – refer to the "Kit Manual" for your particular kit. These instructions describe how to (re)load the software if needed.

Load SNAPpy scripts "demo_sn171.py" and "demo_sn173.py" into the corresponding SNAP Nodes

Note that Portal has to have access to these SNAPpy scripts before it can upload them into your SNAP Nodes. The scripts are located in a subdirectory named "snappyImages" in the source code tree for this example.

Reminder – you can manually copy all of the files in this example's "snappyImages" directory into *Portal's* "snappyImages" directory, or you can use Portal's **Options: Set Working Directory…** feature to "aim" Portal at this *example's* "snappyImages" directory.

If you choose to copy the files, be sure to copy *all* of them (don't forget batmon.py, nv_settings.py, and SN173.py, which get imported by demo_sn171.py and demo_sn173.py).

Load the various "Linux Config" files for this demo onto your E20

Look in the "e20-sys" directory, and follow the instructions in the "readme" file there. Make a note of the directory path in file **e20-kit1-demo.conf** and be prepared to either match it in the next step, or edit that file.

Copy the entire "tree" of files in the web_app directory onto your E20

NOTE – the files must go into a directory tree on the E20 such that the resulting path matches the one specified in **e20-kit1-demo.conf** is correct, or you must manually edit that file.

After you have placed the files, follow the instructions in the "readme" file in that directory too.

Now transition to the "Running This Example" instructions on page 1 of this Quick Start

Either:

A) Power down your E20, and then go to page 1 (where you will power it back up)
or
B) Just reboot your E20 (from the command line) and skip over the part about "powering up the E20" when you go to page 1

Exploring this Example

The web application is a basic Python program built with high-performance libraries, Tornado and SNAP Connect. The Javascript/HTML is kept deliberately simple for ease of understanding, although it showcases a low-latency websockets technique. This can be easily extended to REST interfaces, and other web/backend approaches to fit application requirements. More information about the software used in this example can be found in the "Software" guide (look in the same directory where you found this Quick Start).

See the readme.txt files in the **e20_sys** and **web_app** directories for more details and library dependencies.