



USERS GUIDE

SNAP Connect E10

User Manual for Version 1.1

©2008-2014 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# 600034-01B

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. 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:

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

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 2008-2014, 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

1.	Introduction	1
	Product Overview	1
	Easy-To-Use-Technology	1
	Key Features	1
	Included In the Product	2
	New In This Version	2
	About This Manual	3
	Other Important Documentation	3
	When The Manuals Are Not Enough	4
2.	Network Application Examples	5
	Combining Disparate Networks	5
	Logging Data	8
	Offloading Host Processing	8
	SNAP Lighting From Anywhere In The World	9
3.	SNAP Connect E10 Overview	10
	Product Description	10
	The E10 is a Small Linux Computer	10
	The E10 is a SNAP Node with Serial, Ethernet and USB Interfaces	10
	The E10 Includes an Internal, Wireless SNAP Engine	10
	A Visual Tour	10
	Top View	10
	Wireless Side View	11
	Wired Side View	12
	Bottom View	12
4.	Power and Connectivity	14
	Installing the Drivers	14
	Establishing Connectivity to the E10	14
	Access Via the Serial Port	14
	IP Address Resolution	15
	Setting A Static IP Address	15
	Determining Dynamic IP Addresses	16
	Setting System Hostname	16
	Setting System Time Zone	16

Access Via TCP/IP / Ethernet.....	17
Accessing the SNAP Engine Wirelessly.....	18
5. Extending the E10 with USB Accessories	19
Using usb_modeswitch	19
Connecting to an Additional SNAP Device	20
Establishing a Wi-Fi Connection.....	20
Supported Devices	20
Setting Up Your Wi-Fi Connection	21
Establishing a Data Connection with a USB Cell Modem.....	22
Virgin Mobile MC760.....	23
AT&T USBConnect momentum 4G (Sierra Wireless 313U).....	24
Verizon Pantech UML295	24
6. Common Procedures	27
Editing E10 Files	27
Viewing Other Available Commands	27
Controlling LED A	27
Controlling LED B.....	28
Reading the Mode Button	28
Using an External USB Drive	29
Example 1:	29
Example 2:	29
Setting the E10 to Automatically Update Scripts.....	29
Converting DOS and Windows Text Files.....	29
Creating Custom Python Software.....	30
Making Custom Software run Automatically at Startup.....	30
Method 1: UserMain.py.....	30
Method 2: /etc/inittab	30
Shutting Down UserMain.py Manually	31
Starting UserMain.py Manually	31
Restarting UserMain.py Manually	31
Restoring the Original Behavior of Your E10	31
Temporarily Restoring the Original Behavior of Your E10.....	32
Resetting a Lost User Password	32
Recovering Access to the E10's Radio Node.....	32
7. E10 Recovery, Restore, and Upgrade Procedures	34

Beginning the Manual Installation Process.....	34
Necessary Hardware.....	34
Necessary Software	35
Required Files	35
Recovery Steps.....	35
Completing the Manual Installation Process	37
Running the USB Installer Script	41
SNAP Connect Installations or Upgrades	42
Bridge Recovery Files	42
8. Regulatory Information and Certifications	44
RF Exposure Statement.....	44
FCC Certifications and Regulatory Information (USA Only).....	44
FCC Part 15 Class A	44
Radio Frequency Interference (RFI) (FCC 15.105)	44
Labeling Requirements (FCC 15.19)	44
Modifications (FCC 15.21).....	44
Declaration of Conformity	45
Industry Canada (IC) Certification.....	45

1. Introduction

Product Overview

The SNAP® Connect E10 is a rugged and powerful gateway appliance that allows you to connect to and control your SNAP network nodes across TCP/IP networks.

In addition to providing network connectivity, the E10 contains a powerful Linux-based computer to which you can add your own programs for network participation and control.

In other words, the E10's SNAP Connect software, combined with your own Python program, becomes a node in your SNAP network, responding to RPC messages from other nodes in your network, and generating its own procedure calls for other nodes. The E10 can manage a network of nodes, or it can simply act as a gateway, connecting isolated nodes by passing radio traffic through the Internet. It can collect and store data from nodes in one part of your network, process the data, and forward it to a data warehouse, or send reports or alarms through the Internet to anywhere else in the world. In short, the E10 acts as a point of control and connectivity for extending the reach and capabilities of your SNAP network.



Easy-To-Use-Technology

For many applications, the E10 requires little or no setup. The E10 gateway comes pre-configured to:

- Use DHCP to acquire an IP address
- Automatically participate in any SNAP radio network with compatible radio types and configurations (i.e., radio rate, channel, network ID, encryption settings, etc.)
- Automatically accept incoming TCP/IP connections from other devices running SNAP Connect

User configuration is only required if you want to change these default settings, if you want to connect to additional networks over TCP/IP or serial, or if you want to install your own Python application on the node to interact with the rest of your network in customized ways.

Key Features

Key features of the E10 include:

- ARM Linux operating system (kernel 3.2.34)
- BusyBox multi-call binary for system commands
- 32-bit RISC architecture
- 400 MHz CPU
- 256 MB flash, expandable through external USB connections
- 64 MB RAM
- 10/100 Mb Ethernet port
- USB 2.0 (host) port
- Micro-USB serial and power port

- Small form factor
- Rugged metal case
- Internal Synapse SNAP Engine with external antenna

Included In the Product

Your E10 package includes the following:

- SNAP Connect E10 unit
- External antenna
- Power adapter
- Micro USB cable
- Mounting Brackets

New In This Version

The changes between version 1.0 and version 1.1 of the E10 come entirely from the operating system and included software:

- New packages:
 - `monit` – a utility for managing and monitoring processes, programs, files, directories, and file systems
 - `OpenSSH` – an SSH/SecSH protocol suite providing encryption for network services such as remote login or remote file transfer
 - `NTP` – an operating system daemon that sets and maintains the system time of day in synchronization with Internet standard time servers
 - `rsync` – a fast and extraordinarily versatile file copying tool
 - `screen` – a full-screen window manager that multiplexes a physical terminal between several processes
 - `udev` – a utility that provides a dynamic device directory that only contains files for present devices
 - `iptables` – a utility used to establish, maintain, and inspect IP packet filter rules tables in the Linux kernel
 - `htop` – an interactive process viewer
 - `sqlite` – a lightweight SQL database engine
- USB-Serial support (including support for the SNAP Stick 200 and the SN132 SNAPstick)
- Wi-Fi support (through a USB adapter)
- Cell modem support (through a USB adapter)
- Root file system flash increased to 185 MB
- Addition of `/etc/E10Version`
- Addition of `/etc/TZ`
- Addition of an autorun rule for invoking a script from a USB drive on reboot (`/etc/init.d/S77RunUsb.py`)
- Addition of an autorun rule for invoking a script from a network update server on reboot (`/etc/init.d/X88RunNet.py` – disabled by default)
- Modification of `/etc/fstab`
 - `/tmp` is now limited to 32 MB by default
 - Modify `fstab` to change this limit
- Kernel address locations now dynamically determined by installation script

- Bad blocks can cause the boot loader or kernel images to “spill into” one another
- Bad block locations are analyzed at install time to determine optimal installation locations for kernel images and the bootloader
- Script enhancements
 - BridgeRecovery.py – recover an unresponsive bridge node from the E10 with these scripts available from the Synapse User Forum.
 - UserMain.py / SynapseMain.py
 - Added getIpAddress(), an RPC to return any IP address(es) assigned to the E10
 - Bridge interaction – added options to check or set the bridge channel, network ID, etc. from the E10¹
 - E10ExampleFor1.1.py – E10 LED toggles on traffic and responds to updated UserMain.py/SynapseMain.py when they request the address of the bridge node.
- DHCP improvement
 - Interfaces that use DHCP will continue to attempt to get an IP address
- Python upgraded to version 2.7.5

One side-effect of the breadth of changes between the original E10 release and version 1.1 is that any custom code compiled for the original E10 release is likely to require recompiling before it will function in the 1.1 environment.

About This Manual

This manual provides information specifically regarding the SNAP Connect E10, and covers topics like:

- Initial setup
- Port usage (e.g., the micro-USB serial port, Ethernet port, and USB host port)
- Common applications
- Recovery processes that are specific to this product

The author assumes that:

- You have read and understood the SNAP Primer (or are otherwise familiar with the SNAP product line)
- You have installed the Portal software
- You are familiar with the basics of discovering nodes, uploading SNAPpy scripts to them, and controlling and monitoring them from Synapse’s Portal software

Other Important Documentation

The E10 is typically running Synapse’s SNAP Connect software internally. Be aware that SNAP Connect has its own dedicated user manual. You will likely need to refer to that manual to develop your own applications using the SNAP Connect Python library, as much of the fundamental information on SNAP Connect is not repeated here.

All Synapse documentation is available for free download from our dedicated support forum. Please visit <http://forums.synapse-wireless.com> for more information.

¹ The new "E10exampleFor1.1.py" SNAPpy script should be loaded into the E10's bridge node in order for these new functions to behave as desired.

Some of the most relevant documents are:

- SNAP Primer
- SNAP Connect Users Guide
- SNAP Reference Manual
- SNAP Users Guide
- Portal Reference manual
- SNAP Sniffer Users Guide
- SNAP Hardware Technical Manual
- SNAP Firmware Release Notes
- Portal Release Notes

If you acquired standalone modules (instead of buying a kit), you may also want to download the latest version of the EK2100 or EK2500 user guides. These guides are more tutorial in nature and offer hands- on instructions for demonstrations of SNAP hardware and software.

When The Manuals Are Not Enough

The dedicated support forum offers more than just user documentation. In this forum you can also view questions and answers posted by other users, or post your own questions for discussion. The forum also has examples and application notes available for download. The Synapse website also provides downloads of the latest SNAP, Portal, E10, and SNAP Connect software.

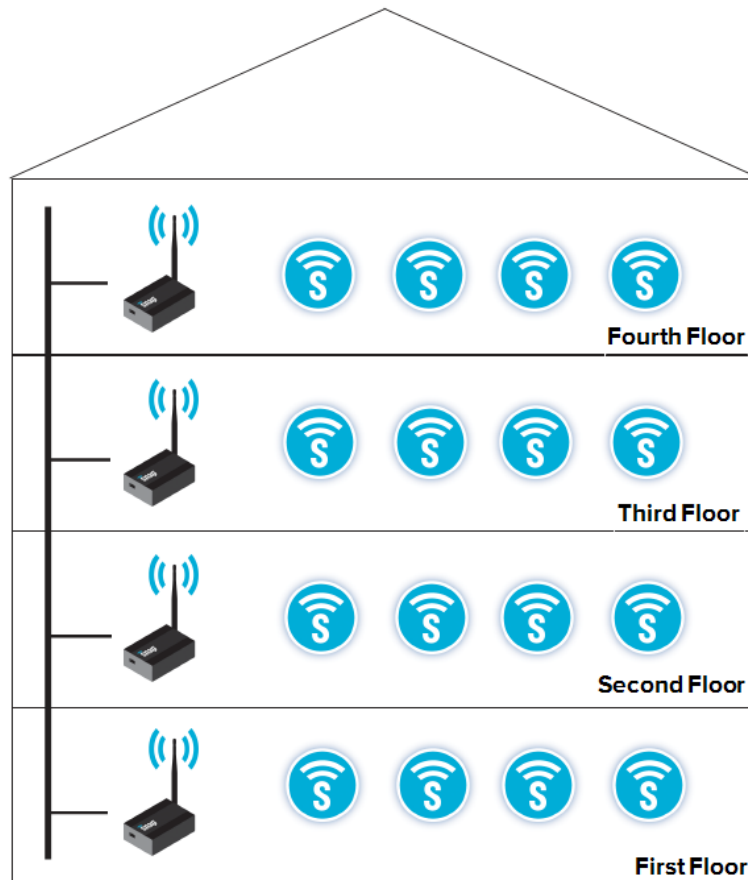
2. Network Application Examples

This chapter offers some high-level examples of how the E10 can be used. Of course, there are just a few examples. Many other network configurations and applications are possible.

Combining Disparate Networks

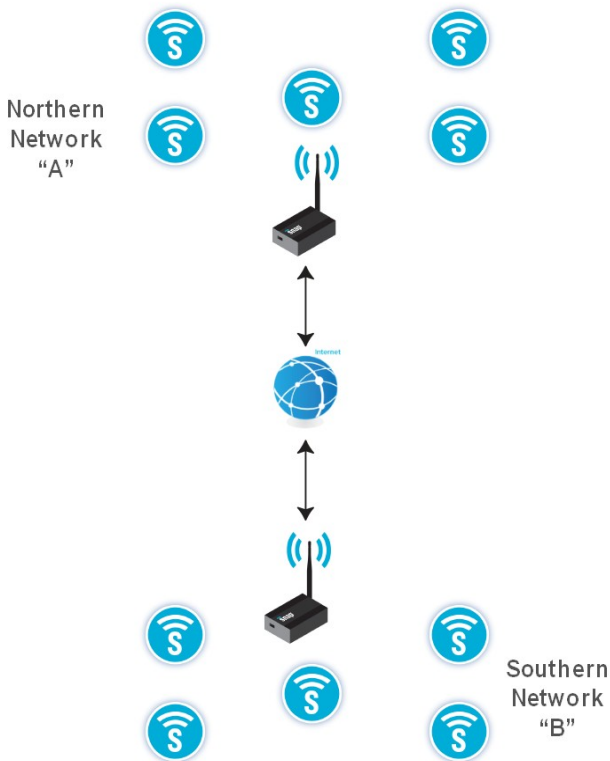
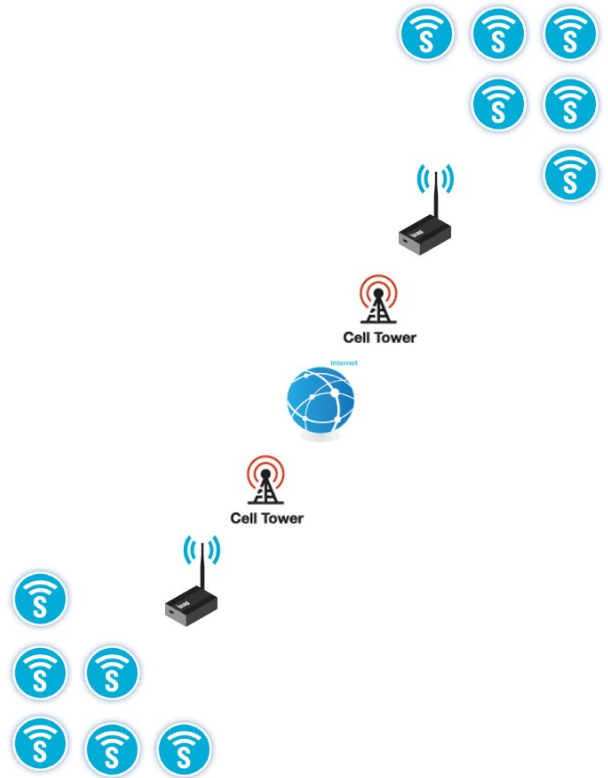
By acting as a bridge between the wireless and Ethernet layers of the network, E10s allow SNAP networks located in separate geographical locations to act as a single large network.

The SNAP subnets being joined might be located in different parts of a building:



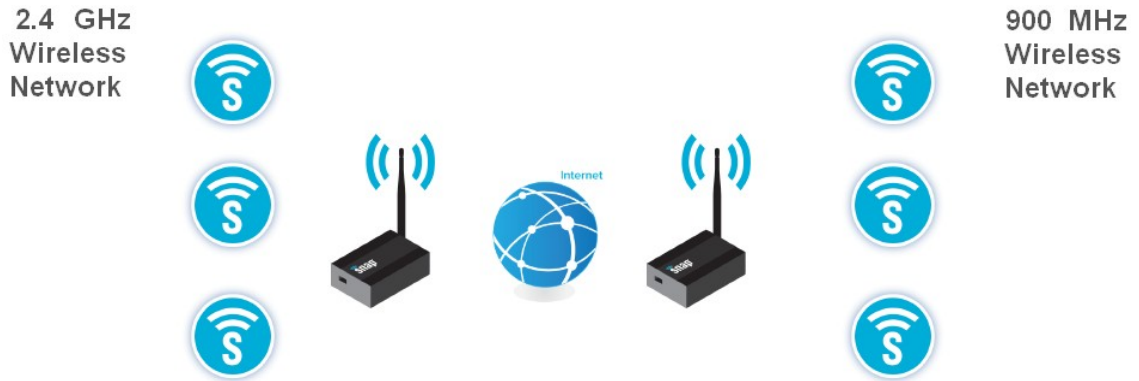
In an environment like this, the subnetworks on different floors might be on different channels in order to minimize cross-talk. Or, it could be that environmental factors prevent a good radio signal from reaching other parts of the network. In either case, an E10 as part of each subnet, linked by the wired TCP/IP network, allows all the subnets to work together as one large network.

Alternately, your problem might be that your subnets are too far apart to be within radio range. If your network has subnets in remote areas, you might find that something as simple as a pair of cellular modems connected to E10s can pass messages on a coordinated scheduled and provide the level of connectivity you need.

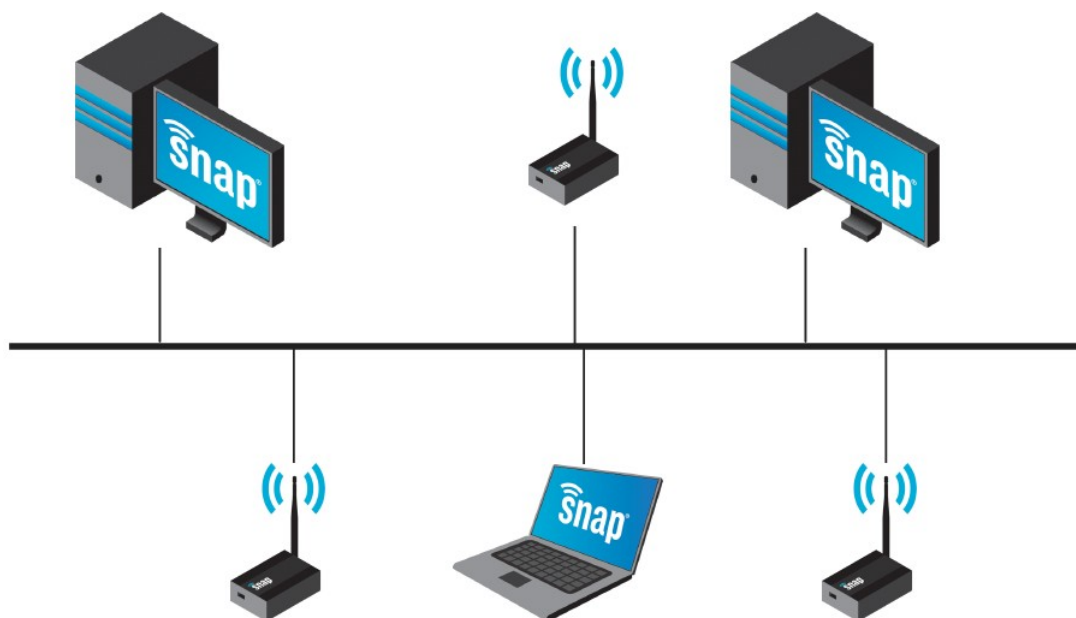


Or, it may be that your subnets are all in established environments where Internet access is always available to provide more constant connectivity than cell modems. If that's the case, E10s connected directly to the Internet can allow all nodes distributed throughout the world to communicate with each other using directly addressed RPC calls.

Because the SNAP Connect gateway is unaware of how any other inter-node communications are handled, you can use a pair of E10s to bridge between two subnets on different configurations of wireless networks. These could be any mixture of subnets running 2.4 GHz radios at any of the available radio rates, along with subnets running 900 MHz radios or subnets running 868 MHz radios, or any combination of these. The bridging performed by the E10s running SNAP Connect overcomes the differences in radio communications before forwarding the packets on to other subnets.



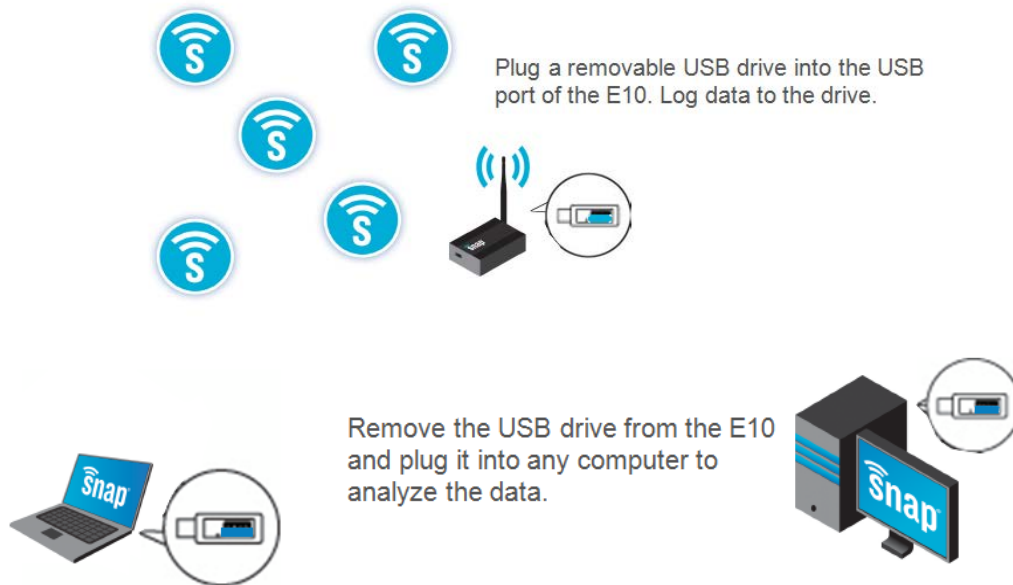
Of course, the E10s are not limited to only communicating with other E10s. The SNAP Connect running on the E10 can make connections, either over TCP/IP or serially, to any other SNAP Connect or Portal instance to which you grant it access.



Refer to the SNAP Connect Users Guide for more information on making connections and controlling other devices.

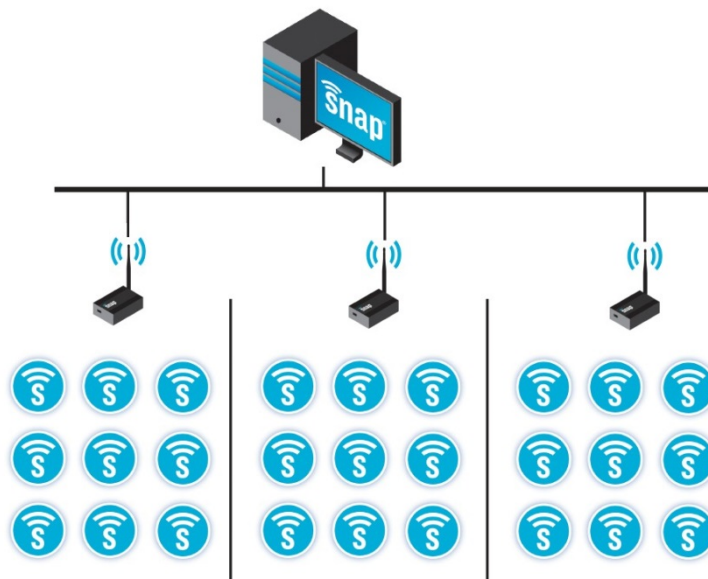
Logging Data

Using the built-in USB host port, an E10 can provide a logging service to a field of SNAP nodes that do not have their own removable USB drives. The logged data can then be analyzed locally or sent (or taken) offsite for more detailed analysis. USB thumb drives or external USB hard drives provide nearly limitless storage options, and the E10 can be configured to upload data (pre-processed or raw) to another host or to the cloud, or the physical media could be collected manually for transfer.



Offloading Host Processing

You can use E10s as an extra tier in larger networks to offload the master computer and move processing closer to the edge. The Python scripts running on the E10s can poll for data, perform data filtering (or other processing), check for alarm conditions, etc.



SNAP Lighting From Anywhere In The World

When combined with Synapse's SNAP Cloud product, E10s support tunneling through firewalls to allow full connectivity to wireless sensor networks from any web browser.

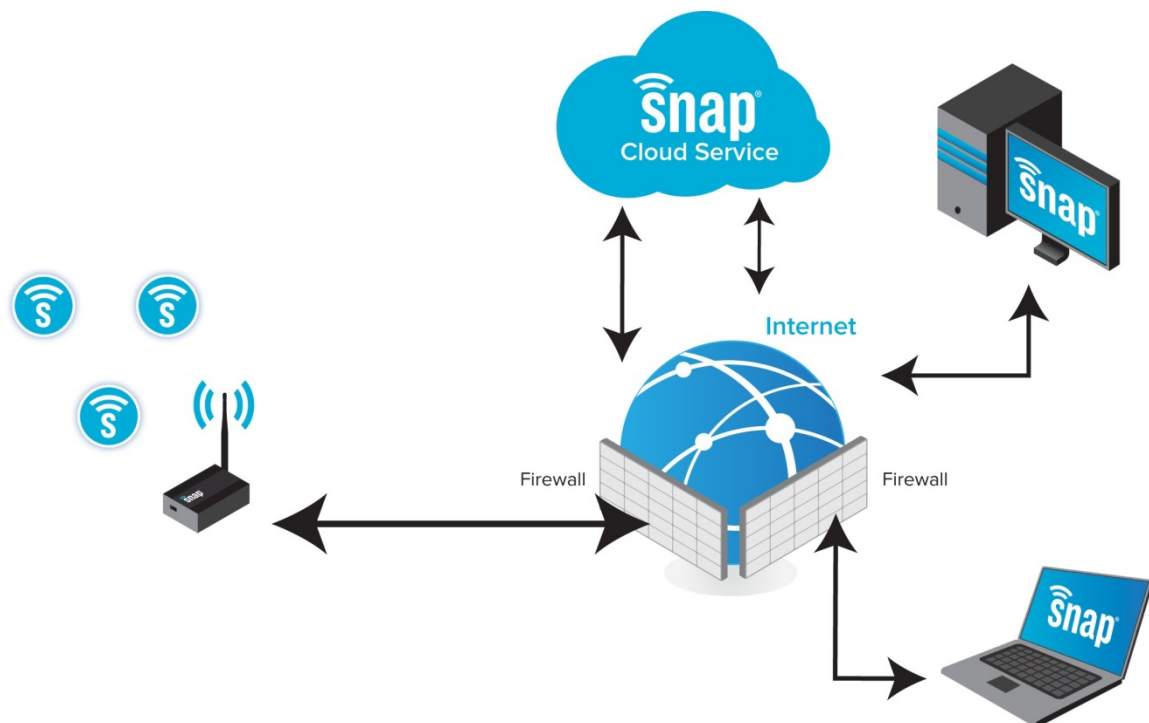
If your E10 is part of a SNAP Lighting kit, there will be a Kit-ID and password on the bottom sticker of the E10. This is your login information for the SNAP Lighting website (at <http://www.SNAPLighting.com>).

Once logged in, you can control and monitor your lighting system via the Internet from anywhere in the world.

This simple-to-use interface is designed to let you remotely perform a number of tasks, such as:

- Turn on/off any of the LED lights on each lighting board included in the kit
- Dim any of the LEDs on any board
- Mix the red, green, blue, and white light intensities
- Read the on-board temperature sensor
- Read the self-calibrating light sensor

NOTE: If your E10 is part of a SNAP Lighting kit, please refer to the SNAPLighting.com Kit User Guide for more information on your product.



3. SNAP Connect E10 Overview

This chapter familiarizes you with the E10, providing a more detailed product description and a visual tour of the product.

Product Description

There are three main things to know about the SNAP Connect E10.

The E10 is a Small Linux Computer

As a Linux computer, the E10 is capable of running custom applications. The system comes with Python installed, and you can install a native ARM-gcc compiler to develop applications in C. If you need further assistance in this area, please contact the Synapse Custom Solutions Group.

NOTE: Advanced Linux users will be interested to know that the E10 is based on the AT91SAM9G20 processor, and is running a variant of ARM-Linux.

The E10 is a SNAP Node with Serial, Ethernet and USB Interfaces

To act as a SNAP Node, the E10 runs Synapse's SNAP Connect software. This gives the E10 the ability to perform mesh routing over the Internet, letting SNAP Nodes in one networked location access a SNAP network in another location.

Also, you can extend the SNAP Connect software (using Python) to add more functions. These can then be invoked via RPC from other SNAP Nodes, much like the functions in SNAPpy scripts. For more information on this feature, refer to the SNAP Connect User guide.

The E10 Includes an Internal, Wireless SNAP Engine

This topic is covered in more detail later in this manual. For now, the important thing to understand is that there are actually two computers contained inside the E10 housing – the ARM processor running SNAP Connect, and the microprocessor on the SNAP Engine itself. They communicate internally using Synapse's Packet Serial protocol over a dedicated serial link.

A Visual Tour

Top View



The top view of the E10 does not reveal much detail. Note that the device is shown without the included mounting brackets that can be used to mount the E10 securely in your operating environment.

The Wireless Side can easily be identified by the RP-SMA antenna mount.

Wireless Side View

The Wireless end of the E10 features the antenna mount, a Mode button, and LED A.



As delivered, LED A glows green when the E10 is powered. Holding the Mode button during startup causes the E10 to bypass the standard SNAP Connect software in the unit. Holding the Mode button for 10 seconds causes the E10 to perform an orderly shutdown.

Having these two purposes for this one button may cause some conflicts. For example if you start holding the button as soon as you apply power to the E10, the 10-second shutdown period expires before the software gets to where it checks the button to see whether it should run `UserMain.py` or `SynapseMain.py`, and the E10 shuts itself down. Furthermore, it is not possible for Synapse to specify a definitive delay period before you should start holding the button because the time it takes to boot your E10 will depend on factors such as your network configuration (Ethernet or Wi-Fi, plus your network speed and the speed at which your NTP server responds).

If you expect to make use of this functionality, configure your environment and then time how long it takes for your E10 to fully boot. The system's check of the button to determine which SNAP Connect script to run is one of the last things to occur during the boot process, at the point when the serial output indicates `Starting user SNAP Connect instance...OK`. Begin holding the button about five seconds before the end of the boot process, and be sure to release the button before you reach the 10-second mark.

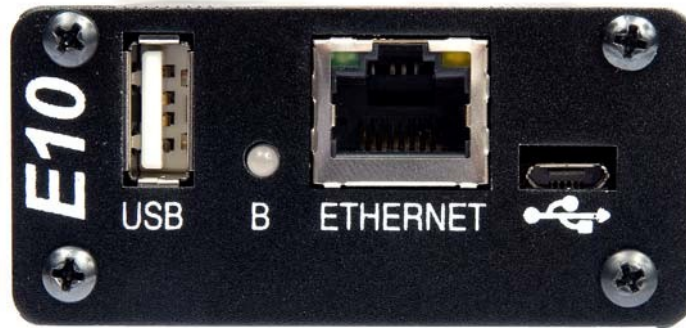
If you have not qualified your boot time this way and do not have a serial connection available to know when your system has completed booting, you can try booting the device and holding the button for about eight seconds, releasing for about two, and repeating until the device has completed booting. This will keep the E10 from performing its 10-second shutdown while still providing a likelihood that the desired script will run.

You can also modify the length of time the button must be held in order to instigate a controlled shutdown. The `/etc/init.d/monitor` file controls this behavior. Set the timeout variable to the number of seconds for which the button must be held if you wish to change the timing, or insert a `#` character at the beginning of the last line (i.e., `#halt`) to disable the button-controlled shutdown entirely.

These LED and button behaviors are the default behaviors when the E10 is shipped. However, both the LED and the Mode button are user-accessible and user-programmable, so you can update these default behaviors to meet your own needs.

Wired Side View


The Wired end of the E10 includes the connectors for wired connections to the device, plus an additional LED, LED B.



The connector on the left, labeled “USB”, is the host USB connection, which you can use to connect an external drive (flash or hard drive), a second radio SNAP node (either a SNAP Stick 200 or an SN132 SNAPstick), or a cellular modem or Wi-Fi dongle. It is not necessary that you connect anything to this USB host port.

By default, the LED (labeled B) glows green when the E10 is initialized and running. This LED, like LED A, is user-controllable so you can change its behavior to provide feedback for your application.

The Ethernet port is your gateway to a TCP/IP network using standard 8P8C (commonly referred to as RJ45) connectors. This lets you connect the E10 to a private network or directly to the Internet for global reporting, networking, or controllability.

Finally, the  logo (representing USB connectivity) marks the Micro-B USB port. This port provides serial connectivity to the E10 (by default at 115,200 baud, 8N1), and also provides power to the E10.

Because this is the power supply for the E10, you must always have this connection in use while the E10 is running. You can have it connected to any host USB port that provides sufficient power, or to a USB power converter, such as the one delivered with the E10.

Bottom View

The bottom of the E10 hosts a sticker that provides information about the device, including the SNAP MAC address of the RF Engine contained within the enclosure, and the SNAP MAC address assigned to the SNAP Connect instance running on the device.

Remember that the E10 actually contains two SNAP nodes. The SNAP Engine (i.e., RF100, RF200, or RF300) that provides radio connectivity is one instance of SNAP, and the SNAP Connect software running in the Linux environment is a separate SNAP Node. Each of these nodes has its own SNAP MAC address (a three-byte hexadecimal address that you may commonly see written in the form D5.AA.96 or ‘\xd5\xaa\x96’). This is the SNAP address you use to unicast a message directly to a node in order to invoke a function on that node by RPC.



Note that the appearance of your sticker may vary; but every E10 will be marked with both SNAP MAC addresses.

If your E10 includes a kit-ID and a password on the bottom, it is part of a SNAP Lighting kit. Please refer to the SNAPLighting.com Kit User Guide for information on your product.

4. Power and Connectivity

It may seem that power and serial connectivity would be entirely separate issues, and warrant separate coverage. However the power for the E10 is provided through the Micro-USB connection (on the “wired” end of the device), which is also where you can make a serial connection to the device.

You can power your E10 using a USB power adapter capable of providing 500 mA of current, such as the one provided with the E10. These devices plug into a wall outlet or power strip and provide the necessary transformation to 5 volts of DC current.

Alternately, you can power your E10 by connecting it to the host USB port of any device that can provide at least 500 mA of current. Typically any port on a PC, laptop or docking station will be able to handle this load. Powered USB hubs can generally do this, but an unpowered hub might not be able to provide the necessary current.

If you choose to connect to a USB port on your computer (either directly or through a powered hub), your computer may prompt you for the location of driver files. The drivers are necessary if you wish to connect serially to the E10. (Most users prefer to have this access.) However, if you only intend to use the computer to provide power to the E10, and you do not need serial access to the device, you can choose to not install the drivers. The USB connection will provide power to the E10 whether the drivers are installed or not.

Installing the Drivers

The drivers you need to install in order to access the E10 serially are available from the website of the USB adapter chip’s manufacturer, at <http://www.ftdichip.com>. You will need the VCP (virtual com port) drivers for the FT2232D chip. FTDI’s website provides an option to search for the latest drivers and download them to your PC. Synapse does not provide the drivers directly in order to ensure that you get the latest drivers available when you need them.

Establishing Connectivity to the E10

You can access the two nodes in an E10 three ways, depending on your network needs. (Remember that the SNAP Engine (i.e., RF100, RF200, or RF30x) in your E10 is one node, while the SNAP Connect instance running on the processor is another node.)

Access Via the Serial Port

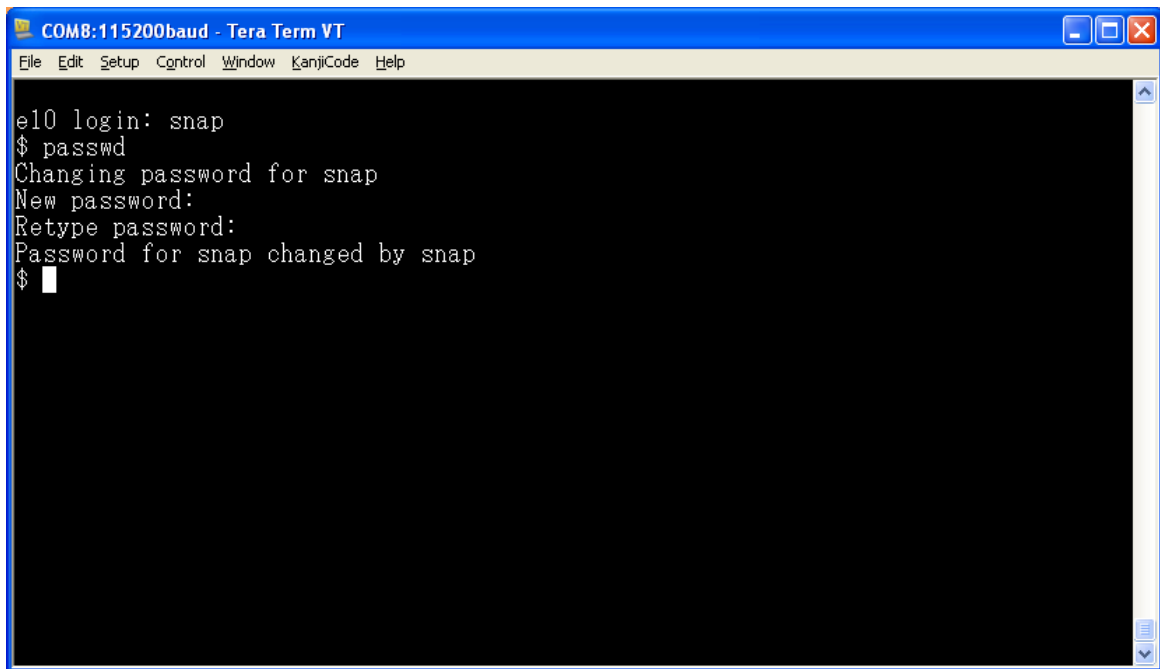
The micro-USB cable provided with the E10 connects to the Micro-B-USB plug on the Wired end of the E10, providing both power and access to the “ttys0” UART of the E10. By default, this port is configured to be a Linux serial console, with the following UART settings:

- 115,200 bps
- 8 data bits
- no parity bit
- 1 stop bit

After you have installed the USB adapter drivers on the PC you are connecting to your E10, the device should be available as a serial connection. On a PC running Windows, you can determine the appropriate COM port by checking the Ports (COM & LPT) section of the Windows Device Manager. This will tell you the COM port to use for connection. (The E10 will also appear as “Synapse USB Device” under the Universal Serial Bus controllers section of the Device Manager.)

Once you have determined the appropriate serial connection, use the communications software of your choice (e.g., Tera Term, RealTerm, PuTTY, CuteCom, etc.) to make your serial connection. Depending on the timing of your connection during the boot process, you may need to press Enter after you connect to receive a login prompt.

E10s ship from Synapse with a default username of `snap`, and with no password. You should immediately change your E10 password on first access, using the Linux `passwd` command. Below is a sample terminal session in which the password is changed.



```
COM8:115200baud - Tera Term VT
File Edit Setup Control Window KanjiCode Help
e10 login: snap
$ passwd
Changing password for snap
New password:
Retype password:
Password for snap changed by snap
$
```

Reminder: You will need to install the FTDI device drivers on your PC before you will be able to make this serial connection.

It will be necessary for you to make at least one serial connection to your E10 before you can make any remote connection to it. This is because remote connections to user accounts without passwords are forbidden, which means you must at a minimum make a serial connection to set the user password (as shown above).

IP Address Resolution

In order to connect remotely, you must know the IP address assigned to the E10. You can determine this over a serial connection, or you can use the serial connection to assign a static IP address.

Setting A Static IP Address

If you wish to assign a static IP address, edit the `/etc/network/interfaces` file on the E10. Locate the following line, and comment it by inserting a “#” at the beginning of the line:

```
iface eth0 inet dhcp
```

Then remove the “#” at the beginning of each of the next four lines to uncomment them, specifying the appropriate address, netmask and gateway values for your network:

```
#iface eth0 inet static
#address x.x.x.x
#netmask x.x.x.x
#gateway x.x.x.x
```

You can edit the `/etc/network/interfaces` file using either nano or vi, both are included by default on the E10. If you are not logged in as root, you will need to use sudo to elevate your user privileges to edit the file.

Determining Dynamic IP Addresses

If you are content to let your E10 be assigned an IP address dynamically, you can use the `ifconfig` command to determine the IP address assigned to the device:

```
# ifconfig
eth0  Link encap:Ethernet      HWaddr 00:1C:2C:00:9D:DA
      inet addr:192.168.2.108  Bcast:192.168.2.255    Mask:255.255.255.0
      UP BROADCAST RUNNING MULTICAST    MTU:1500    Metric:1
      RX packets:1068 errors:1 dropped:0 overruns:0 frame:0
      TX packets:294 errors:0 dropped:0 overruns:0 carrier:0 collisions:0
txqueuelen:1000
      RX bytes:86325 (84.3 KiB)    TX bytes:38280 (37.3 KiB)
      Interrupt:21 Base address:0x4000

lo    Link encap:Local Loopback
      inet addr:127.0.0.1      Mask:255.0.0.0
      UP LOOPBACK RUNNING      MTU:16436    Metric:1
      RX packets:34 errors:0 dropped:0 overruns:0 frame:0
      TX packets:34 errors:0 dropped:0 overruns:0 carrier:0 collisions:0
txqueuelen:0
      RX bytes:2380 (2.3 KiB) TX bytes:2380 (2.3 KiB)

#
```

Note that in the second line of the `eth0` section, the address is shown as `192.168.2.108`. This is the IP address used to make a remote connection to the E10. The IP address assigned to your E10 will likely be different.

Setting System Hostname

For locating and identifying your E10 on your network, you may find it useful to set the device's hostname, which by default is set to "e10". This can be especially useful in an environment where you will have more than one E10 on your network.

To set the hostname for your E10, use nano or vi to edit the `/etc/hostname` file, replacing "e10" with your desired hostname, composed of letters "a" through "z", digits "0" through "9", and hyphens. Spaces and underscores are not permitted. It will be necessary to reboot the E10 for the system to recognize the change to the hostname.

Setting System Time Zone

Version 1.1 of your E10 software automatically queries the Internet (if available) to set the clock on your E10, but it has no way of guessing your current time zone. You can specify this editing the `/etc/TZ` file to include a value from the IANA Time Zone Database. For example, to specify the Central time zone in the United States, you could edit the file to have this line:

```
CST6CDT
```

Access Via TCP/IP / Ethernet

The E10 has a built-in Ethernet interface located on the Wired end of the unit. When a live network cable is plugged into this jack, a green LED (built into the jack – NOT LED B on the E10) glows.

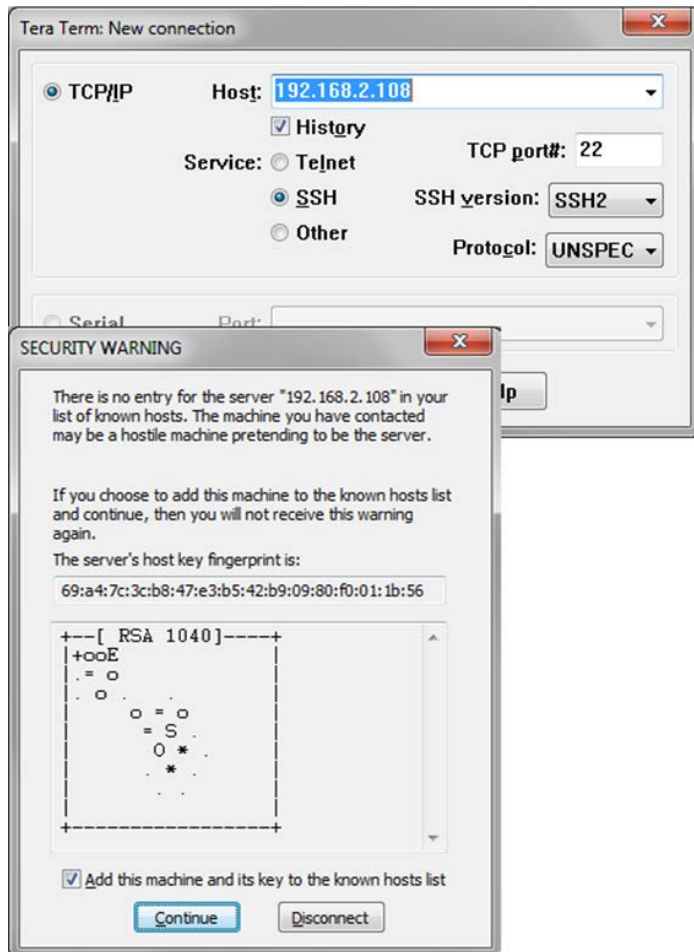
If the E10 detects any Ethernet traffic (even traffic addressed to a different device), a yellow LED on the Ethernet jack blinks.

If you are not getting the expected indicator lights, troubleshoot your network cabling before trying to configure the E10.

Once the E10 is successfully connected to your network, (or the Internet)², you may attempt to log in remotely. To accomplish this, you need to know the E10's IP address.

NOTE: You can only access the E10 over SSH for accounts with passwords. If you have not set a password for the user account (snap, by default, though you can create other user accounts on the system), and if you have not created any other user accounts with passwords assigned, you will need to make a serial connection to the device first and set the user's password.

You might be able to query the routers in your network to determine the IP address assigned to the E10. If not, you will need to make a serial connection to the E10 and use the `ifconfig` command to determine the assigned IP



address, or make a serial connection and set the device to use a static IP address.

Once you have determined the E10's IP address, you can connect to it using any standard SSH client (e.g., Tera Term, RealTerm, PuTTY, etc.) to connect to the device over port 22. The configuration of the login screen varies from application to application, but this example from the open-source Tera Term application gives a fairly typical example.

Your local system's security settings may invoke a warning dialog indicating that you are connecting to an unknown device. You can accept the device as valid and safely connect.

Once you are connected to the E10, you can use the installed TFTP client to access a TFTP server on some other computer to move files to or from the E10. There are many TFTP servers, such as `tftpd32` or `tftpd64`, available from <http://tftpd32.jounin.net/>.

Once you have a TFTP server running as a source (or destination) for files for (or from) your E10, use the `tftp` application on the E10 to transfer your files. Use command `tftp --help` to see the command

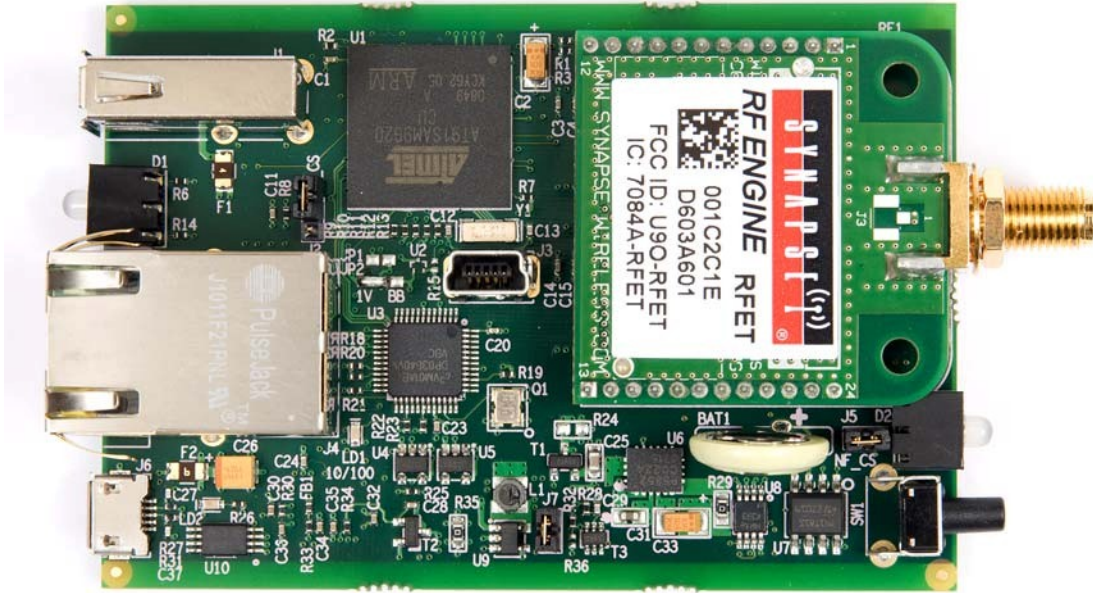
² Note that the E10 can connect to the TCP/IP network either using the Ethernet port or using an appropriately configured Wi-Fi or cell modem device connected to the E10's host USB port. See the next chapter for more information.

line options of the application.

The E10 also comes with an SFTP client installed, which you can use if you need to access files on an FTP server. Use the command `sftp --help` to see the command line options.

Accessing the SNAP Engine Wirelessly

The E10 contains an internal SNAP Engine. The model of SNAP Engine may vary, depending on the specifics of the E10 you have.



You can access this SNAP Engine over the air, (for instance, via Portal), just as you can any other radio node in your SNAP network.

The SNAP Engine runs standard SNAP firmware and can therefore host a SNAPpy script. The SNAP Connect software running on the E10's Linux processor uses this internal SNAP Engine as a bridge to reach the rest of your locally available SNAP network (similar to how Portal uses its own bridge node).

You are not required to install a SNAPpy script on this SNAP Engine, but you should be aware that you can. The SNAP Engine includes a script by default to facilitate access to LED A on the wireless end of the E10. For example, loading the new "E10exampleFor1.1.py" SNAPpy script provided in the E10 1.1.zip file provides these and other functions. The SNAP MAC address of the SNAP Engine in your E10 will be indicated on the sticker on the underside of your E10 enclosure.

NOTE: If you do put a script into the SNAP Engine, be sure to maintain the Packet Serial connection on the second UART (UART 1 for platforms with two UARTS, or UART0, the only UART, on platforms with only one UART) so that the Linux processor continues to have access to the SNAP network.

5. Extending the E10 with USB Accessories

The E10 has drivers to support several USB connectivity devices, such as a second SNAP bridge device (using a SNAP Stick or an SN132), Wi-Fi devices, or cell modems. While the complete details of all configuration options available on these types of devices are outside the scope of this document, we will go over the basics of each.³

Using `usb_modeswitch`

Many USB Wi-Fi and cell modems now come with a small amount of onboard storage, typically used to automatically install drivers when connected to a Windows host. When the device first connects, it appears as a small flash drive or virtual CD-ROM. After installing the necessary drivers, the Windows host sends a signal to the device instructing it to “mode switch” – to unmount the storage and expose itself as a Wi-Fi (or cellular) device.

The Linux `usb_modeswitch` utility allows you to perform this mode switching manually. The utility comes with a number of rules already generated for known devices, so many devices will be converted for you on connecting. If your device does not convert automatically, you can use `usb_modeswitch` to perform the conversion for you.

For example, the MC760, a USB cell modem discussed later in this document, is not immediately recognized and converted. Instead it appears as a CD-ROM. We can use `lsusb` to get the vendor ID and product ID that the device is reporting:

```
Bus 001 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 001 Device 010: ID 1410:5031 Novatel Wireless
```

`usb_modeswitch` typically takes at least four parameters:

- `-v` – the current vendor ID
- `-p` – the current product ID
- `-V` – the desired vendor ID
- `-P` – the desired Product ID

Additionally, some devices will require other arguments, such as a special message to be sent to the device to switch it, supplied with the `-M` parameter. A little Internet sleuthing tells us that the vendor ID is correct at 1410, but the desired product ID is 6002, and that this device requires that we send a hexadecimal message of: 5553424312345678000000000000061b000000020000000000000000000000 to the device to reset it. Thus the full command to send to the device is:

```
usb_modeswitch -v 1410 -p 5031 -V 1410 -P 6002 -M
5553424312345678000000000000061b000000020000000000000000000000
```

Performing this operation returns the following text to the terminal session:

³ Because the E10 is, itself, a USB-powered device, it may not be able to fulfill high power requirements for some USB-connected devices plugged into it. If you find you are experiencing difficulties with your USB device (e.g., Wi-Fi devices losing connectivity or external hard drives failing to mount), try connecting your device to the E10 through a powered USB hub to ensure that the device is receiving sufficient power to drive it.

```
usb 1-1: GSM modem (1-port) converter now attached to ttyUSB0
option 1-1:1.1: GSM modem (1-port) converter detected
usb 1-1: GSM modem (1-port) converter now attached to ttyUSB1
option 1-1:1.2: GSM modem (1-port) converter detected
usb 1-1: GSM modem (1-port) converter now attached to ttyUSB2
scsill : usb-storage 1-1:1.4
```

These messages indicate that the device has now appeared to the E10 as a modem and attached itself properly. A quick check of `lsusb` confirms that:

```
Bus 001 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 001 Device 010: ID 1410:6002 Novatel Wireless
```

Connecting to an Additional SNAP Device

Using the USB port, the E10 can support a second SNAP node. You can connect an SS200 SNAP Stick, you can connect a SNAP Engine using an SN132 SNAPstick device, or you can use an FTDI USB-serial cable to connect to an SN171 ProtoBoard, an SN111 End Device Board, or some other hardware that uses a DE9 connector to make an RS232 serial connection. This can allow your E10 to act as a bridge between two radio subnets, where radios are on some combination of different frequencies, different network IDs, and/or different channels.

The E10 provides the drivers that support the FTDI USB-serial cable and the SN132 inherently. To use the SNAP Stick SS200, you need to modify it to operate as a serial device rather than a USB device. For instructions on doing this, refer to the Synapse Wireless application note “Configuring SS200 SNAP Sticks as COM Ports,” available from the Synapse support forum.

Whichever device you use, simply plug the device into the E10’s host USB port and (among other messages) you should see something similar to:

```
usb 1-1: FTDI USB Serial Device converter now attached to ttyUSB0
```

Or:

```
usb 1-1: cp210x converter now attached to ttyUSB0
```

The key here is the line that says “converter now attached to `ttyUSB#`” (`ttyUSB0`, in the example shown). You will use this device handle, “`ttyUSB#`”, to communicate with the SNAP device. In your SNAP Connect application, you would open a connection to the device like this:

```
com.open_serial(type=SERIAL_TYPE_RS232, '/dev/ttyUSB0')
```

Establishing a Wi-Fi Connection

The E10 comes with driver modules installed for several popular Wi-Fi chipsets. While the drivers for these chipsets have been installed, there are no guarantees that all devices that use these chipsets will work. Synapse has tried to include support for a wide variety of common devices, but we cannot possibly test every USB Wi-Fi adapter on the market to ensure that drivers behave as expected.

Supported Devices

Because many Wi-Fi chipsets do not have direct vendor support in Linux, it is an unfortunate fact that many Wi-Fi adapters may not perform as well as they will in other operating systems where the vendor is more involved in the production of the drivers. We have included a large number of drivers for various Wi-Fi chipsets; however we encourage you to test a new adapter thoroughly before deploying it.

In our testing, we have verified the compatibility of the following devices with the E10:

- Belkin F5D8055
- Linksys WUSB600N

The full list of supported Wi-Fi chipset drivers included in this release is:

- Atheros 7010 / 9271
- Ralink rt2500 / rt2501 / rt61 / rt73 / rt27xx / rt28xx / rt30xx
- Realtek 8192 / 8712 / 8712U / 8192SU / 8187 / 8187B / 8192CU / 8188CU
- Marvell 8xxx Libertas
- Atmel at76c503 / at76c505 / at76c505a
- ZyDas ZD1201 / ZD1211 / ZD1211B
- RNDIS devices

Note that this is the list of chipsets supported. For each chipset, there will be a number of devices that use it.

Setting Up Your Wi-Fi Connection

Fully establishing and configuring a Wi-Fi connection in a Linux environment is a topic that could spawn complete shelves of books. Certainly, exhaustive information about the topic is beyond the scope of this manual. We can provide this brief overview for users new to establishing Wi-Fi connections in Linux, though.

When you plug in your USB Wi-Fi adapter, you may see a message in the console similar to:

```
usb 1-1: new full-speed USB device number 5 using at91_ohci
usb 1-1: reset full-speed USB device number 5 using at91_ohci
usbcore: registered new interface driver rt2800usb
```

The third line, “registered new interface driver rt2800usb”, indicates that the device was recognized as a USB Wi-Fi device using the rt2800usb driver. However, wlan0 (by default, the first Wi-Fi connector, similar to eth0, the first wired Ethernet connection) will not appear in a call to `ifconfig` until we tell the system to look for it.

Start by using `nano` or `vi` to edit the `/etc/network/interfaces` file. (You will need to use the `sudo` command prefix to edit this file if you are not logged in to the E10 as root.) You will see this line in the file:

```
#iface wlan0 inet dhcp
```

Uncomment the line by removing the pound sign at the beginning to enable the E10 to dynamically acquire an IP address.⁴

Next, edit the `/etc/init.d/S40network` file. Uncomment the following line:

```
#WIFI_INAME=wlan0
```

Save your changes to the `/etc/init.d/S40network` file and close it.

The last thing you need to configure for your Wi-Fi connection is the network SSID and, if appropriate, the security key. In the `/etc` directory you will find three files with the name pattern `wpa_supplicant.[encryption_type].conf`, one each for WEP, WPA, and open (unencrypted) networks. Copy the file appropriate for the encryption type of your network over the existing `wpa_ supplicant.conf` file:

⁴ If you wish to define a static IP address instead of having the E10 acquire one dynamically, instead uncomment the `iface wlan0 inet static` line, and the address, netmask, and gateway lines, filling in the appropriate information.

```
sudo cp /etc/wpa_supplicant.wpa.conf /etc/wpa_supplicant.conf
```

After copying, edit the `/etc/wpa_supplicant.conf` file using `nano` or `vi` to specify the network SSID and encryption key (if your network is not open).

Any time you make changes to any of these files, you will need to restart the network services to ensure your changes are applied. You can do that using the following command:

```
sudo /etc/init.d/S40network restart
```

Alternately, you could reboot the device with the `sudo reboot` command. If you reboot by power-cycling the device without first using the `sync` command, your file changes could be lost due to Linux buffering writes to the file system in order to avoid exhausting flash write cycles.

The E10 is configured so that if you switch between different models of Wi-Fi adapters, each new device will be renamed to `wlan0` automatically, so there should be no need to redefine connections to use `wlan1` or other network names. However changing Wi-Fi devices will require a reboot or restart of the `S40network` file.

Establishing a Data Connection with a USB Cell Modem

The E10 provides the drivers for several popular USB cell modem chipsets. Most cell modem drivers communicate over USB-serial rather than specifically as a cell modem device. There is also a generic USB-Serial driver that will work with some devices not specifically listed. The following device lines are installed as loadable modules:

- CP210x
- FTDI
- Motorola Phone modem
- Prolific 2302
- Sierra Wireless
- TI 3410/5052
- Qualcomm
- Qualtech SSU-100

Additionally, there are drivers in place to handle devices that identify themselves as Ethernet-over-USB devices:

- Generic USB-Ethernet
- Sierra Wireless USB-to-WWAN

As with Wi-Fi, it is not practical (or even possible) for Synapse to test every device with the E10, and we cannot guarantee the functionality of a third-party device with a particular chipset with these drivers. We have specifically tested the following devices successfully:

- Virgin Mobile MC760
- AT&T USBConnect Momentum 4G (Sierra Wireless AirCard 313U)
- Verizon Wireless 4G LTE Pantech UML295

The steps required for establishing a connection to a USB cell modem can vary greatly from one modem to another. For example, the MC760 and AirCard 313U both require the use of PPP chat scripts, while the UML295 abstracts this away from the user, simply acting as a router by providing the user a private IP address and transparently handling all the details of connecting to the outside world by itself.

A complete investigation into writing PPP scripts or other connection procedures that your device may require is outside the scope of this document. We will, however, walk through the connection of the three devices we have listed as a template for you going forward.

Virgin Mobile MC760

We briefly discussed the MC760 in the section on using `usb_modeswitch`, noting that the device does not automatically switch modes from storage medium to cellular device.

When you first connect the MC760, it appears to Linux as a virtual CD-ROM drive:

```
usb 1-1: new full-speed USB device number 3 using at91_ohci
scsi0 : usb-storage 1-1:1.0
scsi1 : usb-storage 1-1:1.1
scsi 0:0:0:0: CD-ROM      Novatel      Mass Storage      1.00 PQ: 0 ANSI: 2
scsi 1:0:0:0: Direct-Access Novatel      MMC Storage      2.31 PQ: 0 ANSI: 2
sd 1:0:0:0: [sda] Attached SCSI removable disk
```

Checking `lsusb`, we obtain the vendor ID and product ID presented by the device. If the device is identified with an ID of 1410:5031, it will be necessary to use `usb_modeswitch` to set the proper product ID for the device to be recognized as a USB cell modem. The complete command for the conversion is:

```
sudo usb_modeswitch -v 1410 -p 5031 -V 1410 -P 6002 -M
555342431234567800000000000000061b000000020000000000000000000000
```

Performing this operation returns about 60 lines of text to the terminal session, including this near the end:

```
usb 1-1: GSM modem (1-port) converter now attached to ttyUSB0
option 1-1:1.1: GSM modem (1-port) converter detected
usb 1-1: GSM modem (1-port) converter now attached to ttyUSB1
option 1-1:1.2: GSM modem (1-port) converter detected
usb 1-1: GSM modem (1-port) converter now attached to ttyUSB2
scsi11 : usb-storage 1-1:1.4
```

These messages indicate that the device has now appeared to the E10 as a modem and attached itself properly. A quick check of `lsusb` confirms that change:

```
Bus 001 Device 001: ID 1d6b:0001
Bus 001 Device 010: ID 1410:6002
```

Note that the E10 has now attached the MC760 as three devices in `/dev`: `ttyUSB0`, `ttyUSB1`, and `ttyUSB2`. A `ttyUSB` device represents a USB-to-serial connection. We will use `ttyUSB0` to communicate with the modem and establish a connection using two scripts: a PPP script and a chat script.

Rather than going into detail about PPP here, we refer you to the wealth of documentation available online. The Linux Documentation Project is an excellent source of information, and has an exhaustive PPP-HOWTO document, available at <http://www.tldp.org>.

We have provided the scripts for the MC760 (and other devices we have confirmed) on the E10 for you to use as a template when generating your own scripts. To determine the exact parameters you will need for your script, you will need to either contact the manufacturer of your device or research your device on the Internet.

With our PPP and chat scripts in place, we can run the PPP script to establish a connection:

```
pppd call bb2go &
```

The `bb2go` script establishes a connection with the dialer script, `bb2go_chat`. The `pppd` script should begin working in the background, and should now establish a connection, visible under `ifconfig` as `ppp0`:

```

ifconfig ppp0
ppp0 Link encap:Point-to-Point Protocol
      inet addr:x.x.x.x P-t-P:x.x.x.x      Mask:255.255.255.255
      UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
      RX packets:6 errors:0 dropped:0 overruns:0 frame:0
      TX packets:7 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:3
      RX bytes:72 (72.0 B)      TX bytes:141 (141.0 B)

```

You should now have an active internet connection.

To end the connection, kill the `pppd` execution process. You can determine which process is appropriate by searching through the currently running processes for a `pppd` process:

```
ps | grep pppd
```

This will return a list of all processes invoked with the `pppd` keyword, including the `grep` process compiling the list. Look for a result something like this:

```
2090 root pppd call bb2go
```

The first column is the process number, and will almost certainly be different for your call. End the process with a `kill` command:

```
kill 2090
```

AT&T USBConnect momentum 4G (Sierra Wireless 313U)

Connecting the 313U works effectively the same as the MC760. However you will not need to use `usb_modeswitch` first. The configuring of the device should happen automatically. The following output should let you know that the Sierra Wireless drivers detected the 313U and attached to five `ttyUSB` devices:

```

sierra_net 1-1:1.7: wwan0: register 'sierra_net' at usb-at91-1, Sierra
Wireless USB-to-WWAN Modem, xx:xx:xx:xx:xx:xx
usbcore: registered new interface driver sierra_net
USB Serial support registered for Sierra USB modem sierra 1-1:1.0: Sierra USB modem
converter detected
usb 1-1: Sierra USB modem converter now attached to ttyUSB0
sierra 1-1:1.1: Sierra USB modem converter detected
usb 1-1: Sierra USB modem converter now attached to ttyUSB1
sierra 1-1:1.2: Sierra USB modem converter detected
usb 1-1: Sierra USB modem converter now attached to ttyUSB2
sierra 1-1:1.3: Sierra USB modem converter detected
usb 1-1: Sierra USB modem converter now attached to ttyUSB3
sierra 1-1:1.4: Sierra USB modem converter detected
usb 1-1: Sierra USB modem converter now attached to ttyUSB4
usbcore: registered new interface driver sierra
sierra: v.1.7.16:USB Driver for Sierra Wireless USB modems

```

From there, it's a matter of calling the PPP script in `/etc/ppp/peers/` written for the 313U:

```
pppd call attMomentum &
```

This should, as with the Virgin Mobile MC760, result in a data connection being established on interface `ppp0`. Also, as with the MC760, you will need to explicitly kill the `pppd` process in order to end your data connection.

Verizon Pantech UML295

The UML295 operates much differently from the MC760 or the 313U. Rather than writing and calling PPP scripts, the UML295 operates as a gateway device between the E10 and the cellular connection. Several seconds after connecting, the E10 should produce the following output (or similar) to its terminal:

```
usb 1-1: new full-speed USB device number 10 using at91_ohci
scsi8 : usb-storage 1-1:1.0
scsi 8:0:0:0: CD-ROM    PANTECH    CD-ROM    0000 PQ: 0 ANSI: 2
```

After several seconds, the UML295 will switch modes, providing an Ethernet connection to the E10:

```
usb 1-1: USB disconnect, device number 10
usb 1-1: new full-speed USB device number 11 using at91_ohci
generic-usb 0003:10A9:6064.0001: claimed by neither input, hiddev nor hidraw
generic-usb 0003:10A9:6064.0002: claimed by neither input, hiddev nor hidraw
cdc_ether 1-1:1.0: eth1: register 'cdc_ether' at usb-at91-1, CDC Ethernet
Device, xx:xx:xx:xx:xx:xx
usbcore: registered new interface driver cdc_ether
```

The UML295 has now registered itself as an Ethernet device on eth1. (This could be different if you already have some other device using interface eth1.) If it does not do this automatically, simply call:

```
sudo ifconfig eth1 up
```

We can request an IP address for eth1 from the UML295 with udhcpc:

```
sudo udhcpc -ieth1
```

This should generate text something like the following:

```
udhcpc (v1.20.2) started
Sending discover...
Sending select for 192.168.32.83...
Lease of 192.168.32.83 obtained, lease time 604800
deleting routers
route: SIOCDELRT: No such process
adding dns 192.168.32.2
```

In this case, the E10 understands itself to have been assigned IP address 192.168.32.83. (Your address may vary.)

Interacting with the UML295 now requires using HTTP post requests. The UML295 exposes several configuration pages that can be obtained and examined with wget:

```
wget http://192.168.32.2/condata -O /tmp/condata
```

Note that you will use the IP address specified as the DNS IP address, not the IP address assigned to the E10. This generates text like the following:

```
Connecting to 192.168.32.2 (192.168.32.2:80)
condata 100% |*****| 3273 0:00:00 ETA
```

You can view the /tmp/condata file with vi or nano (or cat) to get an idea of what we post to it. We are more interested in the actions section, which shows we can use this page to tell the UML295 to connect:

```
wget http://192.168.32.2/condata?action=connect -O /tmp/condata
```

That generates the following text⁵:

```
Connecting to 192.168.32.2 (192.168.32.2:80)
condata 100% |*****| 3273 0:00:00 ETA
```

The green light on the UML295 should come on, and if you wget condata again, you can parse through the XML to view your external IP address, if interested.

To disconnect, post again, but this time with 'disconnect':

⁵ Using the -O switch to direct the file to /tmp/condata is optional, though the resulting file can be parsed for potentially useful information. If you wish to post the connect or disconnect action without generating an output file, you can use the -s switch instead: wget -s http://192.168.32.2/condata?action=disconnect

```
wget http://192.168.32.2/condata?action=disconnect -O /tmp/condata
```

That should provide the following output:

```
Connecting to 192.168.32.2 (192.168.32.2:80)
condata      100% |*****| 3273 0:00:00 ETA
```

The device should disconnect. It should be noted that even while disconnected, eth1 will show your local IP (e.g., 192.168.32.83) provided by the UML295. This is the address that the UML295 uses internally for its communications with the E10, and does not mean that the E10 is available to any broader network at that IP address.

6. Common Procedures

This section contains short example procedures for some common E10 functions. For more information on these topics, please refer to the standard Linux documentation.

Editing E10 Files

The E10 comes with the popular vi and nano editors installed. As root, or as another user with appropriate permissions to edit a file, you can edit configuration or data files on the E10 using commands like:

```
nano filename
vi filename
```

System files are typically owned by `root`, so you are likely to need to use the `sudo` command prefix to elevate your permissions in order to edit them:

```
sudo nano /etc/hostname
sudo vi /etc/hostname
```

CAUTION: Because of the way the Linux file system buffers writes to the flash memory to avoid exhausting the write cycles, files you have modified and closed might not actually be committed to the file system immediately after you edit them. This is generally performed transparently to you and you shouldn't worry about it, unless you frequently remove power to the device without doing a clean shutdown. In addition to flushing the buffers periodically during runtime, the file system will automatically finalize all changes to the flash if you reboot or shutdown the device. If you remove power without shutting down gracefully, however, this never has a chance to occur, and you could find that you have lost work. If you absolutely must remove power without shutting down, call `sync` from the command line to force the buffers to flush to the flash before doing so.

Viewing Other Available Commands

The E10 uses the popular BusyBox multi-call binary. To see which commands are built into BusyBox, try the command: `busybox --help` from the command line.

Additional commands available on the operating system can be found in the `/bin`, `/sbin`, `/usr/bin`, and `/usr/sbin` directories.

Controlling LED A

LED A is on the wireless end of the E10. It is controlled by the internal SNAP Engine using GPIO pins `GPIO_0` and `GPIO_1`. This is actually a tri-color LED, controlled as follows.

	GPIO_0 True	GPIO_0 False
GPIO_1 True	LED Amber	LED Red
GPIO_1 False	LED Green	LED Off

As shipped, the internal SNAP Engine is loaded with a SNAPpy script that lights the LED green when the unit is powered. This SNAPpy script also provides the following functions that can be invoked via RPC:

- `setLedAOff()`
- `setLedAGreen()`

- `setLedARed()`
- `setLedAYellow()`

If you load your own application script into the SNAP Engine you can implement your own controls of `GPIO_0` and `GPIO_1` to control the LED, but the four `setLedA[Color]()` functions will not be available to you.

Controlling LED B

LED B is on the wired end of the E10. It is controlled by the Linux processor. By default, this LED glows green once the E10 has completed its boot cycle.

As with LED A, this is a tricolor LED that can yield green, red, or amber (if both green and red are on). Try these commands⁶ from the Linux command line:

```
# For green LED
echo 1 > /sys/class/gpio/gpio76/value
echo 0 > /sys/class/gpio/gpio76/value
# For red LED
echo 1 > /sys/class/gpio/gpio77/value
echo 0 > /sys/class/gpio/gpio77/value
```

There are also shortcuts in place in `/usr/bin`:

- `greenled 0`
- `greenled off`
- `greenled 1`
- `greenled on`
- `redled 0`
- `redled off`
- `redled 1`
- `redled on`

If you have not changed the `UserMain.py` file that comes loaded on the E10, you can control LED B from your SNAP network by RPC as well:

- `setLedBOff()`
- `setLedBGreen()`
- `setLedBRed()`
- `setLedBYellow()`

Examine the `UserMain.py` file to see how these functions work.

Reading the Mode Button

The Mode button on the wireless end of the E10 is connected to the E10's Linux processor. It can be monitored via the contents of the virtual file `/sys/class/gpio/gpio74/value`:

```
cat /sys/class/gpio/gpio74/value
```

This command⁷ displays a 1 when the button is not pressed, or a 0 when the button is pressed.

⁶ These commands are different from those in the first version of the E10, though the callable functions in the `UserMain.py` have maintained the same names and signatures. Echoing values to `/sys/class/leds/greenled/brightness` or `/sys/class/leds/redled/brightness` will no longer work.

⁷ This command is also different from the underlying infrastructure in the first version of the E10, though the `/usr/bin/button` command from the first version, and the `readButton()` function in `UserMain.py`, have remained constant. The `gpio9260` program available in 1.0 still works, but it deprecated.

A simpler way to read the Mode button is to use the button program that comes with the E10:

```
/usr/bin/button
```

This program returns 0 if the button is pressed, and returns a non-zero value if the button is not pressed.

Assuming you have not changed the default `UserMain.py` file, the button can also be read via SNAP RPC by calling function `readButton()`.

Using an External USB Drive

Upon detecting that a USB flash drive has been inserted, the operating system will attempt to mount the drive automatically and make it accessible via `/mnt/usb`.

If the automatic mounting fails, you can generally mount the flash drive manually. The exact commands needed to mount different USB drives vary between drives, but here are some examples.

Example 1:

```
mdev -s  
mount /dev/sda /mnt
```

Example 2:

```
mdev -s  
mount -t vfat /dev/sda1 /mnt
```

After executing the correct command, you should be able to find your USB drive files under the `/mnt` directory.

Setting the E10 to Automatically Update Scripts

You can set your E10 to automatically check for updated applications on reboot. The applications can be delivered on a USB drive or by an update server to which the E10 makes a network connection.

By default, the `/etc/init.d/S77RunUsb.py` rule is enabled on the E10. This rule causes the E10 to mount an inserted USB drive when it boots and, if it can find one, run the Python script named `RunMe.py` located in the drive's root. You can edit the `/etc/init.d/S77RunUsb.py` rule to modify this behavior, or rename the file (e.g., to `/etc/init.d/X77RunUsb.py`) to disable the feature.

There is also a rule file named `/etc/init.d/X88RunNet.py` that you can modify to connect to a network server and download a script to run when the E10 boots. This rule is disabled by default. In order to enable the rule, rename the file to `/etc/init.d/S88RunNet.py` and edit the file to reference the appropriate server and file to download. By default, the script downloads from a URL specific to the E10's MAC address, allowing you to specify different update scripts for different E10s.

Converting DOS and Windows Text Files

You may sometimes need to clean up text files that came from a DOS or Windows system in order to use them on the E10's Linux system. Windows and DOS use different characters to terminate lines in text files from those that Linux uses, which may cause files to behave strangely if they originated in a different file system.

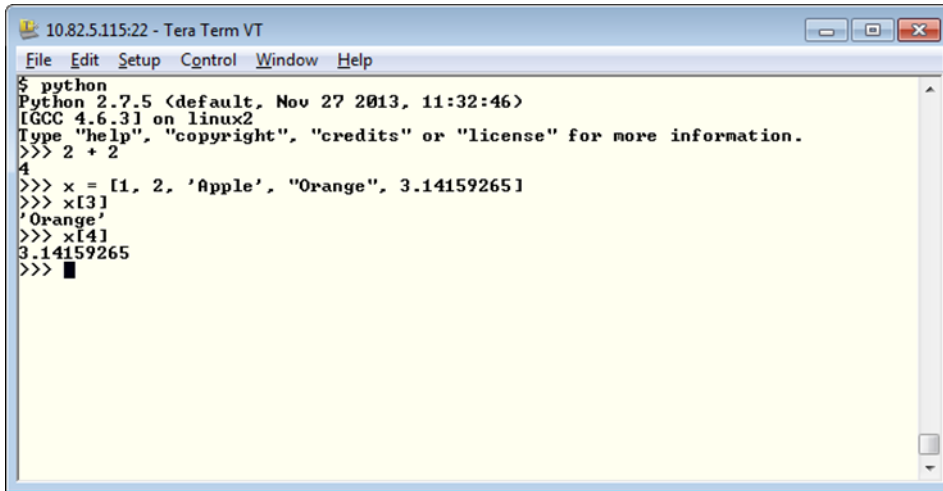
If you transfer a file to your E10 (for example, some Python source code) and it seems to have a lot of funny characters (^M) at the end of some or all lines, try entering this command:

```
dos2unix <file_to_be_repaired>
```

where `<file_to_be_repaired>` is the path and filename of the text file to correct.

Creating Custom Python Software

This procedure shows you how to create your own custom Python software. The E10 comes with Python 2.7.5 already installed. You can write and execute your own Python programs. You can also use Python in interactive mode:



```
10.82.5.115:22 - Tera Term VT
File Edit Setup Control Window Help
$ python
Python 2.7.5 (default, Nov 27 2013, 11:32:46)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> 2 + 2
4
>>> x = [1, 2, 'Apple', 'Orange', 3.14159265]
>>> x[3]
'Orange'
>>> x[4]
3.14159265
>>> █
```

Most of the standard Python modules are available for use in your code, with the following exceptions:

- `_tkinter`
- `bsddb1`
- `dbm`
- `Gdbm`
- `imageop`
- `nis`
- `sunaudiodev`

Making Custom Software run Automatically at Startup

There are two ways you can invoke your own software when the E10 boots. Only one SNAP Connect application at a time can connect to the RF Engine in the E10, though you could have a second SNAP Connect application working through a SNAP node connected to the E10's USB port.

Method 1: `UserMain.py`

The E10 comes pre-configured to automatically launch the Python file `/home/snap/UserMain.py` during system startup, so the easiest thing to do is just modify this source file to do what you want. You can see how Python gets launched by looking in the `/etc/init.d/S999snap` file.

Method 2: `/etc/inittab`

You can also invoke your own program on startup by adding a line similar to the following to your

`/etc/inittab` file:

```
null::sysinit:<path to program> <arguments to program>
```

For example:

```
null::sysinit:/user/bin/python /home/snap/UserMain.py
```

Shutting Down UserMain.py Manually

If you need to halt `UserMain.py` (for example, you are going to make changes to it), use the following command:

```
sudo /etc/init.d/S999snap stop
```

Starting UserMain.py Manually

To manually restart `UserMain.py` without rebooting the E10, you can use the following command:

```
sudo /etc/init.d/S999snap start
```

Restarting UserMain.py Manually

To stop and then restart `UserMain.py`, you could use the previous two commands, or you could perform the following single command:

```
sudo /etc/init.d/S999snap restart
```

Restoring the Original Behavior of Your E10

If you have made customizations to your `UserMain.py` and want the original functionality back, be aware that file `SynapseMain.py` (also located in `/home/snap`) is a backup copy of the `UserMain.py` file installed when you received the E10. If you overwrite `UserMain.py` with a copy of `SynapseMain.py`, the original functionality will be restored (and all your customizations to `UserMain.py` will be lost).

Obviously this also doesn't apply if you have modified or removed the `SynapseMain.py` file. We recommend that you leave this file in place to allow the device to be restored in the future.

Of course, this only restores the functionality of the `UserMain.py` script. If you have made other changes to your E10 as discussed here (to `/etc/inittab` or `/etc/hostname` or `/etc/wpa_supplicant.conf`, for example) those changes will not be undone by restoring your `UserMain.py` file.

Also, any configuration changes you have applied by means of NV parameters (such as enabling encryption) will not necessarily be cleared as a result of replacing your `UserMain.py` file. To reset all your factory NV parameters, you can delete the `nvparams.dat` file in `/home/snap`. The next time SNAP Connect runs it will recreate this file with default parameters. This may cause problems, though, as SNAP Connect typically runs as root when the E10 is booted, and the `snap` user will no longer have appropriate permissions to edit the file. To prevent this, make sure the `snap` home directory is your working directory and use the following commands to stop the current instance of SNAP Connect, remove the parameters file, and create a new file as the `snap` user:

```
sudo /etc/init.d/S999snap stop
sudo rm nvparams.dat
python UserMain.py
```

Now when you reboot the E10, SNAP Connect will use the parameters file created by the `snap` user, and the `snap` user will still have full read/write authority to affect the file.

Temporarily Restoring the Original Behavior of Your E10

The above procedure permanently erases any changes you have made to the `UserMain.py` file. If you just want to run the `SynapseMain.py` code for a single instance, hold down the Mode button while you reboot the E10 and throughout the boot process. Doing so causes `SynapseMain.py` to launch instead of `UserMain.py`.

Obviously this doesn't apply if you have modified or removed the `SynapseMain.py` file. However, if you set aside an additional copy of `SynapseMain.py` in case you wish to restore it later and then modify both your `UserMain.py` and `SynapseMain.py` files for different purposes, you can use this trick of holding the Mode button during reboot to invoke your own code for special startup circumstances. (Remember the caveat related to holding this button discussed in the Wireless Side View section in the Power and Connectivity chapter.)

Note that rebooting the E10 while holding the button does not affect any NV parameters that have been set on the device, so you may still have encryption enabled, or various networking features disabled or modified, depending on the system's configuration, unless you have explicitly modified your `SynapseMain.py` file to set those NV parameters to some known state.

Resetting a Lost User Password

If you lose your user password you can interrupt a device boot, edit the boot arguments of U-Boot, and boot into the device without a password in order to reset the password. Follow this procedure:

1. Boot the E10 and quickly make a serial connection to it during the boot process.
2. When the STDOUT text tells you to "Hit any key to stop autoboot" press a key and the E10 will boot into the U-Boot environment instead of the E10 Linux environment.
3. From the `U-Boot>` prompt, enter `edit bootargs` and press Enter.
4. Add the word `single` to the end of the line of boot arguments and press Enter.
5. From the `U-Boot>` prompt, enter `boot` and press Enter.
6. The E10 will boot into its Linux environment as the root user. Use the `passwd` command to set the password for the user of your choice (typically `snap`).
7. Use the `reboot` command to reboot the E10 into its normal environment. There is no need to re-edit the boot arguments in the U-Boot environment.

Recovering Access to the E10's Radio Node

There are many ways a user might lose access to an RF Engine radio node. For example, the script loaded into the node might drop the node into an infinite loop, or an improperly set (or forgotten) encryption key might render the node inaccessible over the air and over the serial connection.

With the 1.1 version of the E10 software, you can install a collection of scripts for recovering access to such a node. Download the `BridgeRecovery.zip` file from the Synapse User Forum and transfer the file to your E10. Unzip the file in your home directory. Then the `/home/snap/BridgeRecovery/` directory will contain a Python script named `FlashBridge.pyc`, (along with several supporting scripts), that allows you to remove the script from an unresponsive RF Engine, or to clear the node's NV parameters. (If the node's script explicitly sets NV parameters, you may need to clear the script and then reset the NV parameters.) You can use the following command (from the `BridgeRecovery` directory) to get a list of arguments for the script:

```
python FlashBridge.pyc -h
```

For example, to remove the `SNAPpy` script from an RF100 RF Engine in your E10, you would use these commands:

```
sudo resetBridge.sh  
python FlashBridge.pyc -rf100 -e -p /dev/ttyS1
```

You can also use the `FlashBridge.pyc` script to upgrade the firmware in your SNAP Engine. Copy the firmware file (which ends with a `.sfi` file suffix) to your E10 using TFTP or a USB device, and then use the `--image` switch to upload the firmware image.

7. E10 Recovery, Restore, and Upgrade Procedures

From time to time, Synapse releases updated versions of both the underlying operating system and the SNAP Connect software library. If you are logged into your E10 with root-level privileges (or as a user who can be elevated to root-level privileges), you can upgrade your E10 when new software is available.

If you are unable to boot your E10 into U-Boot due to some system corruption, you will need to perform a manual recovery of the device as described in the Beginning the Manual Installation Process section, below. From there you can continue with the manual installation process, or you can run the Linux installer from a USB drive connected to your E10.

Even if your E10 is functioning you can still perform the installation manually. Most users will find it easier, however, to skip all the manual steps and run the installer, which will replace version 1.0 with version 1.1, or will restore your 1.1 installation to its factory state.

In either case, performing the OS level upgrade completely replaces the contents of your E10 with new files in a new file system. Any configuration you have performed, and any program or data files on your E10 will be lost as part of this upgrade. Also, once you have recovered the E10 to the 1.1 version of the Linux system, you will need to use the instructions in the SNAP Connect Installations or Upgrades section to install SNAP Connect on the device in order to use the gateway in a SNAP network.

Beginning the Manual Installation Process

The manual installation process is required if your E10 is unable to boot to U-Boot due to some unknown system corruption. If your E10 is able to boot to U-Boot (even if it cannot boot all the way into Linux), it will be easier to skip ahead to the Running the USB Installer Script section.

You will need to download the E10 1.1 installer, currently available from the Synapse Wireless user forum. The downloaded installer file will have a name like `E10v1.1.3_Recovery.zip`. (The exact version number, “1.1.3” in this example, may vary for your download. Adjust your expectations accordingly. For the remainder of this process, this file will be referred to as the recovery zip file.) Unzip it in the location of your choice on your computer (Windows, Mac, or Linux). This will create an `E10v1.1.3_Recovery` directory with a collection of files and folders in it. (Depending on the tool you use for unzipping, the collection of files may be nested in an additional `E10v1.1.3_Recovery` directory.)

Beginning a recovery installation this way restores only U-Boot and the recovery kernel on the E10. Once this is done, users will be able to run the installation script to complete the process, or may continue with the manual installation process. In either case, this full restoration replaces the operating system and file system, removing anything that had previously been on your E10.

Necessary Hardware

- You will need a screwdriver with the correct tip to remove the screws on the end of your E10. Depending on your E10, this may be a Phillips head screwdriver or a Torx T-10 screwdriver.
- A USB cable with a standard Type A plug on one end and a Mini B plug on the other.

NOTE: This is different from the cable provided with your E10, which includes a Type A plug on one end and a Micro B plug on the other. (You will also need that cable.)

- You may also need a pair of needle-nose pliers to remove and reinstall a jumper if you do not have small fingers.

Necessary Software

All the files and software needed for the recovery process can be found in the E10 recovery zip file available on the Synapse User Forum. The E10 is based on an ATMEL chipset, so an ATMEL tool, included in the recovery zip file, is used to push the data files to the E10. This program, named SAM-BA CDC, runs on Windows. There is not a version of this program for Macintosh or Linux computers.

Required Files

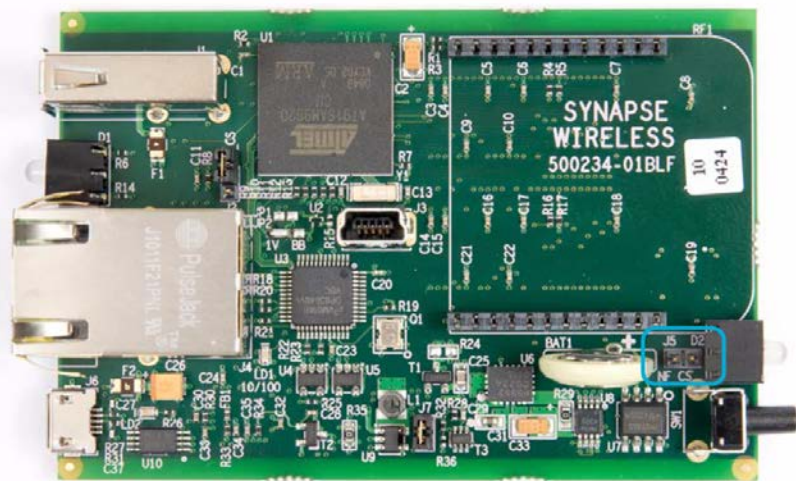
The following files are also necessary for this process, and can be found in the E10 recovery zip file as well:

- bootstrap.bin
- u-boot.bin
- uImage-primary.bin
- uImage-recovery.bin

Recovery Steps

Step 1: Remove power from the E10 by unplugging the USB Micro B plug from the “wired” end of the E10. Remove any devices connected to the USB host, and remove the Ethernet cable, if one is connected.

Step 2: Open your E10 by removing the four screws from the “wireless” end of the E10, the end that has the antenna connection on it. Gently grasp the E10 “wireless” end cap and slide the circuit board straight out of the metal enclosure.



Step 3: Remove jumper J5. You may need needle-nose pliers to remove the jumper that resides directly behind the LED mounting next to the SNAP Engine on the “wireless” end of the E10. You might find it helpful to remove the SNAP Engine first for easier access to the jumper, as shown in this picture.

Step 4: Power up the E10 by using the micro USB cable provided with your E10 to connect the device to your Windows PC. Wait at least three seconds before continuing.

Step 5: Reinstall jumper J5 after the step 4 delay. (It is still ok to not have the SNAP Engine in place.)

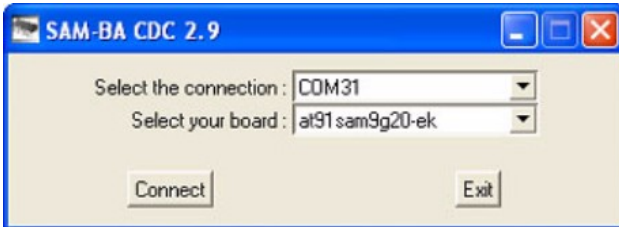
Step 6: Connect the mini USB cable to your PC and to the E10. Remember that this is a different USB cable from the one provided with your E10, and that you must have both USB connections in place to perform this recovery upgrade.

If your PC prompts you for the location of a driver file when you make this



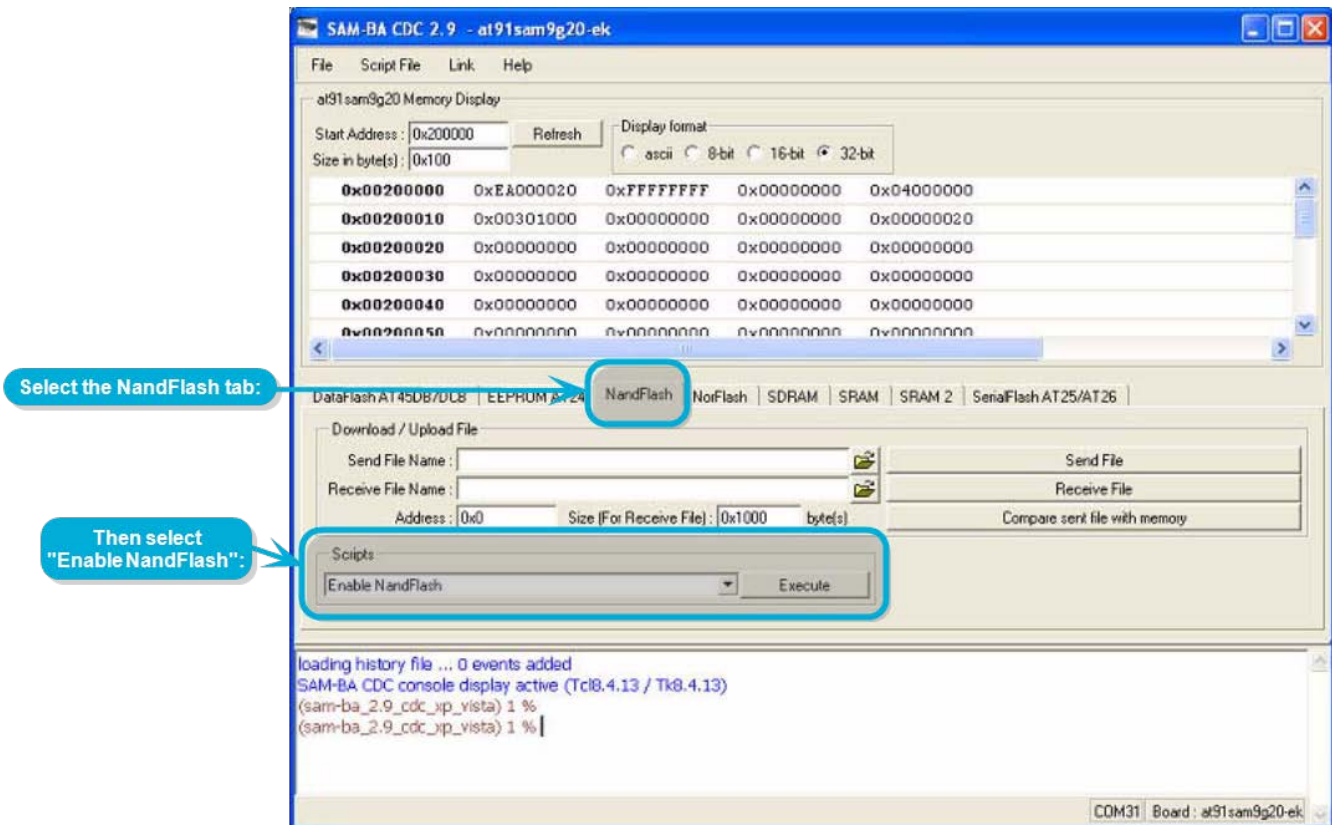
connection, direct the driver installer to the `drv` directory inside the `sam-ba_2.9_cdc` directory of the files you extracted from the E10 recovery zip file.

Step 7: Launch the SAM-BA CDC program. You may receive a Windows dialog asking if you wish to run a program from an unknown publisher. If you do, click the Run button.



Step 8: Make the connection. You should see a dialog box asking you to select the proper serial connection and the appropriate board, as in this dialog. The correct connection will vary depending on your computer configuration, but you should select the connection that formed when you connected the USB mini connection,

not the USB micro connection that you normally use to power the board and make a serial connection. For the board, select `at91sam9g20-ek`, which corresponds to the ATMEL processor used in the E10. After selecting the serial connection and board, click the Connect button.



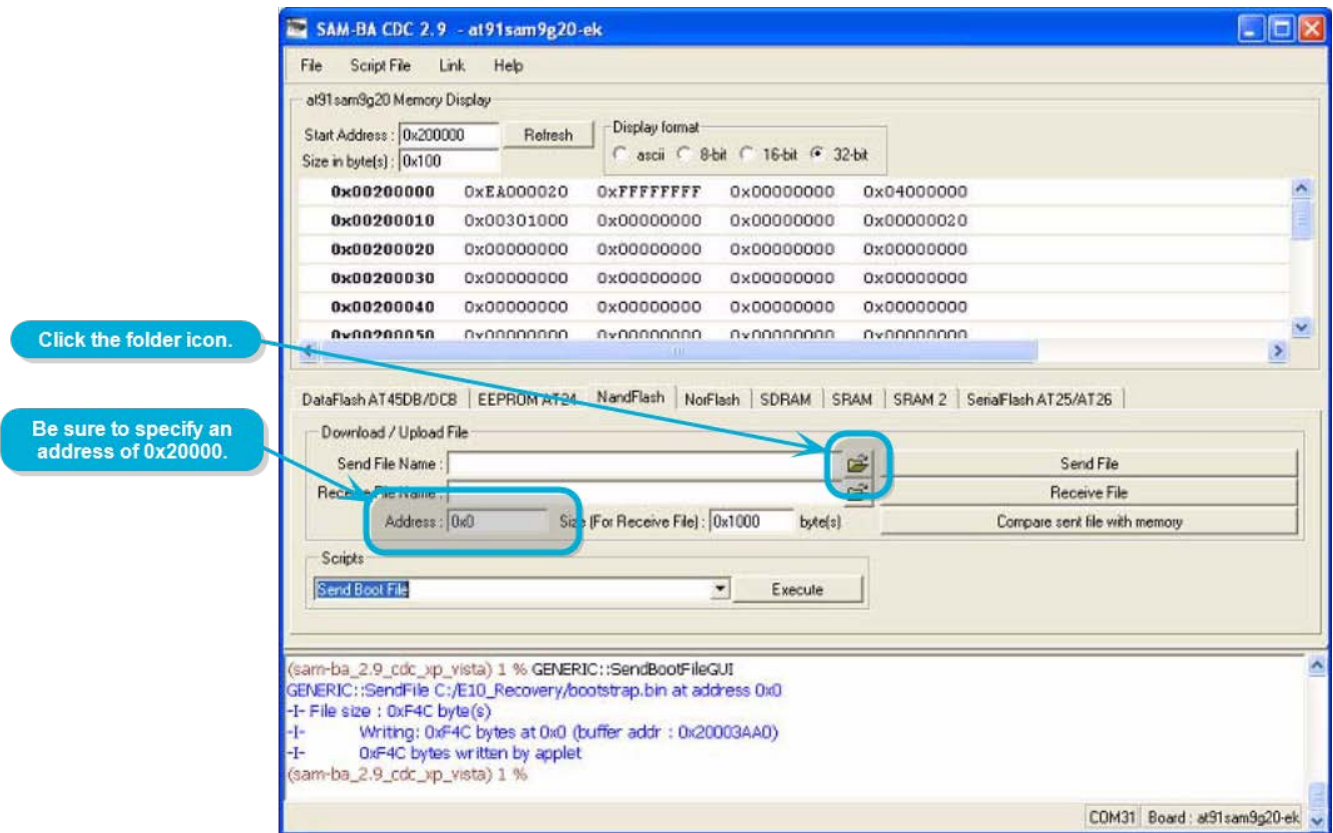
Step 9: Enable the NandFlash. When the main screen displays, select the NandFlash tab (about in the middle of the screen), and then select “Enable NandFlash” in the Scripts selection box. Click the Execute button.

Step 10: Erase the current flash by selecting “Erase All” in the Scripts selection box and clicking the Execute button.

Step 11: Send the bootloader file to the E10 by selecting “Send Boot File” in the Scripts selection box and clicking the Execute button. You will be prompted to browse for the appropriate file to send. Browse to the directory unzipped from the E10 installer zip file and select the `bootstrap.bin` file. Click the Open button.

Step 12: Send the remaining files to the E10 using the “Download / Upload File” section just above the Scripts selection box. Click the folder icon at the right edge of the “Send File Name” field to browse for the file to send

to the E10. Browse to the directory unzipped from the E10 recovery zip file, and select the `u-boot.bin` file. In the Address field in that section, specify an address of `0x20000`. (All of those characters that look like zeros are zeros, not capital O's.) Click the Send File button.



The SAM-BA CDC program will transfer that file to the E10, which may take a few seconds. When it completes, repeat this process with file `uImage-primary.bin`, sent to address `0x120000`, and file `uImage-recovery.bin`, sent to address `0x3c0000`. Some of these files will take longer to send than others (`uImage-recovery.bin` being the largest), but each transmission is complete when the send dialog closes.

Step 13: Shut down the E10 by removing the micro USB connection and the mini USB connection. Reattach the RF Engine if you previously removed it. Reassemble the E10 into its case by inserting the board into the second slot from the bottom (not the lowest slot), and reattaching the “wireless” end plate with the screws removed in Step 2.

At this point U-Boot and the recovery kernel are installed and the E10 can be booted. You can skip ahead to the Running the USB Installer Script section. Otherwise, if you wish for any reason to perform a manual installation, continue with the next section describing the manual installation of the primary file system.

Completing the Manual Installation Process

Before you perform the steps in this section, you must perform the steps in the previous section, Beginning the Manual Installation Process. Perform the steps in this section only after that section has been completed.

Step 1: Connect the E10 to your PC and quickly make a serial connection to the device using the E10’s Micro B USB connection (8N1, 115200 baud) using the terminal program of your choice. Watch for the text `Hit any key to stop autoboot:` early in the process and press a key (such as the space bar) to divert the E10 into its built-in U-Boot firmware manager.

NOTE: If you take too long to make the serial connection and miss the opportunity to stop the autoboot, the system boots to Linux recovery kernel installation. If this happens, log in as `root` with the password `recovery` and enter the command `reboot now` to reboot the device. Watch for the `Hit any key to stop autoboot:` message and respond appropriately.

Instead of a normal Linux prompt you should see a U-Boot prompt that looks like this:

```
U-Boot>
```

Step 2: Check the sticker on your E10's enclosure to determine the device's MAC address. This will be a 12-character (six-byte) hexadecimal number of the format `001C2Cuvwxyz`, and should not be confused with the SNAP MAC address, which will be 16 characters (eight bytes), but which also starts with `001C2C`. When you have determined the MAC address, enter the following command at the `U-Boot>` prompt:

```
setenv ethaddr 00:1C:2C:uv:wx:yz
```

where `uv:wx:yz` are the characters from the MAC address on the sticker. (Once set for an installation, the MAC address becomes immutable for the device. If for some reason the MAC address has already been set on the device, you will receive the message `Can't overwrite "ethaddr"`. If this occurs, skip ahead to Step 5. You can use the command `printenv ethaddr` to determine the value set for the MAC address.)

Step 3: Enter the following command, which will generate no output:

```
saveenv
```

Step 4: Enter the following command:

```
reset
```

This will cause the E10 to reboot. You should monitor the output as the device boots, and again press a key to stop the autoboot process and enter the U-Boot firmware manager.

Step 5: From the `U-Boot>` command prompt, issue the command `run bootrecovery` to boot to the recovery kernel.

Step 6: Wait for the E10 to reach the recovery Linux login prompt, and use the username `root` and password `recovery` to log in.

Step 7: If you need to specify a static IP address for your E10, use the `ifconfig` command to do so now. For example, to manually set your IP address to `192.168.8.8`, you would use this command:

```
ifconfig eth0 192.168.8.8
```

Step 8: Enter the following command:

```
flash_eraseall /dev/mtd4
```

This will produce the following output on the screen:

```
Erasing 128 Kibyte @ 20e0000 -- 100 % complete
```

Step 9: Enter the following command:

```
flash_eraseall /dev/mtd5
```

This will produce several lines of output indicating that blocks are erased and that the command is skipping bad flash blocks. Your output will vary depending on the condition of the flash in your device, but you should end with this line:

```
Erasing 128 Kibyte @ cd20000 -- 100 % complete
```

Step 10: Enter the following command:

```
ubidetach -p /dev/mtd5
```

This will produce the following output to the screen:

```
UBI: mtd5 is detached from ubi0
```

Step 11: Enter the following command:

```
ubiformat -y /dev/mtd4
```

This will produce several lines of output ending with one like this:

```
ubiformat: formatting eraseblock 1641 -- 100 % complete
```

Step 12: Enter the following command:

```
ubiattach -p /dev/mtd4
```

This will generate about 25 lines of output, ending with one like this:

```
UBI device number 0, total 264 LEBs (34062336 bytes, 32.5 MiB), available 258 LEBs (33288192 bytes, 31.7 MiB), LEB size 129024 bytes (126.0 KiB)
```

Step 13: Enter the following command:

```
ubimkvol -m /dev/ubi0 -N recoverytmp
```

That will produce the following output:

```
Set volume size to 33288192
Volume ID 0, size 258 LEBs (33288192 bytes, 31.7 MiB), LEB size 129024 bytes (126.0 KiB), dynamic, name "recoverytmp", alignment 1
```

Step 14: Enter the following command, which will generate no output:

```
mkdir /root/recoverytmp
```

Step 15: Enter the following command:

```
mount -t ubifs /dev/ubi0_0 /root/recoverytmp
```

This will produce the following output:

```
UBIFS: default file-system created
UBIFS: mounted UBI device 0, volume 0, name "recoverytmp"
UBIFS: file system size: 32126976 bytes (31374 KiB, 30 MiB, 249 LEBs)
UBIFS: journal size: 1548288 bytes (1512 KiB, 1 MiB, 12 LEBs) UBIFS: media format:
w4/r0 (latest is w4/r0)
UBIFS: default compressor: lzo
UBIFS: reserved for root: 1517435 bytes (1481 KiB)
```

Step 16: Enter the following command, which will generate no output:

```
cd /root/recoverytmp
```

Step 17: Use TFTP or a USB drive to transfer the `rootfs.ubifs` file from the E10 recovery zip file to the `/root/recoverytmp` directory on the E10. (TFTP transfers require that an Ethernet cable be connected to the E10.)

Step 18: Enter the following command:

```
ubiformat -y /dev/mtd5
```

This produces several lines of output, ending with something like the following:

```
ubiformat: formatting eraseblock 1641 -- 100 % complete
```

Step 19: Enter the following command:

```
ubiattach -p /dev/mtd5
```

This will generate about 25 lines of output, ending with one like this:

```
UBI device number 1, total 1637 LEBs (211212288 bytes, 201.4 MiB), available  
1617 LEBs (208631808 bytes, 199.0 MiB), LEB size 129024 bytes (126.0 KiB)
```

Step 20: Enter the following command:

```
ubimkvol -m /dev/ubi1 -N rootfs
```

This generates the following output:

```
Set volume size to 208631808  
Volume ID 0, size 1617 LEBs (208631808 bytes, 199.0 MiB), LEB size 129024 bytes  
(126.0 KiB), dynamic, name "rootfs", alignment 1
```

Step 21: Enter the following command, which generates no output:

```
ubiupdatevol /dev/ubi1_0 rootfs.ubifs
```

Step 22: Use TFTP or a USB drive to transfer the `patches_1.1.x.zip` file and the `applypatches.sh` file from the E10 recovery zip file to the `/root/recoverytmp` directory on the E10. (Your exact file name version will differ if you are installing a later version of the E10 OS. Some releases may not include a `patches_1.1.x.zip` file. If your recovery zip file does not include a `patches` zip file, you may – but do not have to – skip step 26, below.)

Step 23: Enter the following command, which generates no output:

```
rm rootfs.ubifs
```

Step 24: Enter the following command, which generates no output:

```
mkdir /root/rootfs
```

Step 25: Enter the following command, which generates about a half dozen lines of output:

```
mount -t ubifs /dev/ubi1_0 /root/rootfs
```

Step 26: Enter the following command:

```
sh applypatches.sh /root/rootfs
```

If your recovery zip file included a `patches` zip file, this generates several lines of output, ending with the following:

```
IncrementVersionNumber.sh: Success
```

If your recovery zip file did not include a `patches` zip file, you will instead see a message indicating that the `patches_*.zip` file could not be opened.

Step 27: Enter the following command:

```
fw_setenv bootcmd "run bootmain"
```

This will set U-Boot to boot to the primary Linux kernel automatically. On success it produces no output.

Step 28: Enter the following command to perform a controlled shutdown of the system and commit all changes:

```
halt
```

This completes the Linux installation process. You can now disconnect the USB cable powering the E10 and reboot it, which will be necessary before installing SNAP Connect. Remember, having performed the manual installation process, there is no need to follow the instructions in the Running the USB Installer Script section.

Running the USB Installer Script

If your E10 is functional enough to boot to U-Boot, even if your Linux environment is misconfigured or missing, the easiest path to updating your gateway is to install using the USB installer script. You can run the installer on E10s to upgrade them from version 1.0 to the latest 1.1 version, or you can run it to upgrade one 1.1 version to a newer one, or you can run it against a current installation to refresh the E10 back to its factory state.

The USB installer script performs its installation from files you copy to a USB drive and then plug into the E10.

Step 1: First, download the E10 1.1 installer zip file from the Synapse Wireless user forum. The downloaded installer file will have a name like `E10v1.1.3_Recovery.zip`. (The exact version number, “1.1.3” in this example, may vary for your download. Adjust your expectations accordingly. For the remainder of this process, this file will be referred to as the recovery zip file.) Unzip it in the location of your choice on your computer (Windows, Mac, or Linux). This will create an `E10v1.1.3_Recovery` directory with a collection of files and folders in it. (Depending on the tool you use for unzipping, the collection of files may be nested in an additional `E10v1.1.3_Recovery` directory.)

Step 2: Copy all files, but not the `sam-ba-2.9_cdc` folder *nor the containing folder*, to the root directory of a USB drive. You cannot have these files organized in a directory on your USB drive. They must reside at the drive’s root level. (It may be best to either dedicate a USB drive to this purpose, or to temporarily move all other files into a single directory at the drive’s root level in order to keep things tidy and easy to clean up afterward.)

Step 3: Boot your E10 and quickly make a serial connection to it using the micro-USB connection. As the device begins to boot, watch for the text “Hit any key to stop autoboot” early in the output to STDOUT, and press a key so the E10 will boot into the U-Boot environment instead of the E10 Linux Environment. If you miss this text, after the device completes booting you can use the `reboot` (or `sudo reboot`) command to reboot the E10 without losing the serial connection, which will make it easier to catch the interrupt opportunity.

Step 4: At the `U-Boot>` command prompt, enter (or copy/paste) the following text as one line, and press Enter:

```
saveenv; saveenv; set bootargs console=ttyS0,115200 root=/dev/mtdblock1
mtdparts=atmel_nand:128k(bootstrap),1M(uboot),15232k(kernel),120M(rootfs),-(other)
rw rootfstype=jffs2; usb start; fatload usb 0 0x21000000 up-uI-installuboot.bin;
bootm 0x21000000;
```

This will generate output to your terminal session indicating its progress, and will take several minutes to complete. When it is finished, you will see the message:

```
Welcome to SnapConnect e10 v1.1
E10 login:
```

Step 5: When the installation process is complete, you will need to set the MAC address of your E10. Reboot your gateway (you can log in using the `snap` user with no password, and use the `sudo reboot` command to do this) and again interrupt the boot process when you see the “Hit any key to stop autoboot” message.

Step 6: Check the sticker on your E10’s enclosure to determine the device’s MAC address. This will be a 12-character (six-byte) hexadecimal number of the format `001C2Cuvwxyz`, and should not be confused with the SNAP MAC address, which will be 16 characters (eight bytes), but which also starts with `001C2C`. When you have determined the MAC address, enter the following command at the `U-Boot>` prompt:

```
setenv ethaddr 00:1C:2C:uv:wx:yz
```

where `uv:wx:yz` are the characters from the MAC address on the sticker. (Once set for an installation, the MAC address becomes immutable for the device. If for some reason the MAC address has already been set on the device, you will receive the message `Can't overwrite "ethaddr"`. If this occurs, skip ahead to Step 8 to reboot the device. You can then rerun the installation to enable the setting of the address again. If you make a mistake

in setting the MAC address, you must rerun the installation. You can use the command `printenv ethaddr` from the U-Boot prompt to determine the value set for the MAC address.)

Step 7: Enter the following command, which will generate no output:

```
Saveenv
```

Step 8: Enter the following command:

```
reset
```

This will cause the E10 to reboot. When the boot process is complete, the Linux system in place will have a user named `snap`, which has a blank password assigned. The root user will be returned to a disabled state with no password set, but there shouldn't be any need to set it. Remember, it will still be necessary to install SNAP Connect on the E10 before you can use it as a SNAP gateway device. See the next section for instructions on installing SNAP Connect.

SNAP Connect Installations or Upgrades

Use the instructions in this section to install SNAP Connect, to upgrade to a new version of SNAP Connect, or as a template for loading or upgrading your own custom application software.

If you are replacing an existing SNAP Connect installation, you should stop the existing SNAP Connect instance from running, and then remove any SNAP Connect directories that exist on your E10.

```
sudo /etc/init.d/S999snap stop
cd /usr/lib/python2.7/site-packages/ sudo rm -r apy
sudo rm -r serialwrapper
sudo rm -r snapconnect
sudo rm -r snaplib
cd ~
```

New versions of SNAP Connect for the E10 are released as zip files, and as of this writing are available online from the Synapse user forum. Move these files to your E10 `/home/snap` directory using `tftp`, or on a USB drive. Make sure you select the installation for the version of Python installed on the E10, 2.7.5. Once you have the zip file in the `/home/snap` directory, use `unzip` to extract it:

```
unzip -q snapconnect-3.1.0-python2.7.zip
```

After you unzip the file⁸, you must change directory into the newly unzipped file directory and then run the following command:

```
sudo python setup.py install
```

This installation process moves all the appropriate files to their necessary locations. Now when you boot your E10, the `UserMain.py` application (which, by default, is a SNAP Connect application) will run.

Bridge Recovery Files

After you install SNAP Connect, you can also download some bridge recovery files from the forum that allow you to use the SNAP Connect instance to recover access to a bridge device that is locked because of an invalid script or because of misconfigured NV parameters. Download the `BridgeRecovery.zip` file and transfer it to the `/home/snap` directory of your E10. Unzip the file with this command:

⁸ The name of your file may be different from the “`snapconnect-3.1.0-python2.7.zip`” name shown. Adapt your command accordingly.


```
unzip -q BridgeRecovery.zip
```

Refer to the Recovering Access to the E10's Radio Node section for details on using these scripts.

8. Regulatory Information and Certifications

RF Exposure Statement

This equipment complies with FCC radiation exposure limits set forth for an uncontrolled environment. This equipment should be installed and operated with minimum distance of 20cm between the radiator and your body. This transmitter must not be co-located or operating in conjunction with any other antenna or transmitter.

FCC Certifications and Regulatory Information (USA Only)

FCC Part 15 Class A

These devices comply with part 15 of the FCC rules. Operation is subject to the following two conditions: (1) These devices may not cause harmful interference, and (2) These devices must accept any interference received, including interference that may cause harmful operation.

Radio Frequency Interference (RFI) (FCC 15.105)

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation.

If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient or relocate the receiving antenna.
- Increase the separation between the equipment and receiver.
- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
- Consult the dealer or an experienced radio/TV technician for help.

Labeling Requirements (FCC 15.19)

This device complies with Part 15 of FCC rules. Operation is subject to the following two conditions: (1) this device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

If the FCC ID for the module inside this product enclosure is not visible when installed inside another device, then the outside of the device into which this product is installed must also display a label referring to the enclosed module FCC ID.

Modifications (FCC 15.21)

Changes or modifications to this equipment not expressly approved by Synapse Wireless, Inc. may void the user's authority to operate this equipment.

Declaration of Conformity

(In accordance with FCC 96-208 and 95-19)

Manufacturer's Name: Synapse Wireless, Inc.

Headquarters: 6723 Odyssey Drive
Huntsville, AL 35806

Synapse Wireless, Inc. declares that the product:

Product Name: SNAP Connect E10

Model Number: SLE10-001

to which this declaration relates, meet the requirements specified by the Federal Communications Commission as detailed in the following specifications:

- Part 15, Subpart B, for Class A equipment
- FCC 96-208 as it applies to Class A personal computers and peripherals

The products listed above have been tested at an External Test Laboratory certified per FCC rules and has been found to meet the FCC, Part 15, Emission Limits. Documentation is on file and available from Synapse Wireless, Inc.

Industry Canada (IC) Certification

This digital apparatus does not exceed the Class A limits for radio noise emissions from digital apparatus set out in the Radio Interference Regulations of the Canadian Department of Communications.

Le présent appareil numérique n'émet pas de bruits radioélectriques dépassant les limites applicables aux appareils numériques de la class A prescrites dans le Règlement sur le brouillage radioélectrique édicté par le ministère des Communications du Canada.