Recipe for running a py code on NEO

1. Open a terminal window and cd dorjamjr/dirk/

2. Put your py code in this folder (example: yourCode.py)

3. Start an HTTP server with: python -m SimpleHTTPServer

4. Open a SECOND terminal window and type: ssh root@10.0.0.88

5. Enter the password: connectbox

6. In that window, type: wget 10.0.0.58:8000/yourCode.py

7. Run your code: python3 yourCode.py

To download to iMac from github, wget won't work. Use curl instead as:

curl https://raw.githubusercontent.com/GeoDirk/NEO\_LED\_Blink/master/batpowershutdown.py -o batcode.py

(Note above is all on one line.)

**Better**

OK… using the scp command I can move a file named testscp.py from the

NEO /root directory to the local directory (from which I am running the terminal) with:

scp root@10.0.0.74:/root/testscp.py ./

(note that the terminal used for this command is the one on the Mac, not the NEO)

Similarly, I can move the file ./serv.sh from the Mac to the NEO /root directory with:

scp ./serv.sh root@10.0.0.74:/root

**WRITING TO DEVICE MEMORY**

First, install the devmem2 library with:

apt-get install devmem2

Then you can use the command devmem2 <address> [b|h|w] <write value> like:

devmem2 0x01c2081c h 0x2000

**Enabling SSH on recent releases of NEO code**

From: https://github.com/ConnectBox/connectbox-pi/blob/master/docs/deployment.md

sshd is not running on pre-made release images. To permanently enable it, make a directory called .connectbox on your USB storage device and place a file named enable-ssh in that folder. Insert your USB storage into the ConnectBox and power it on. Once the system has booted, you will be able to ssh to the ConnectBox as root/connectbox. Please change the root password immediately.

Note that once the USB stick has been used to enable ssh, subsequent boots will have ssh enabled even if the USB stick isn't present. Also note, it appears that the ip address is remembered on a NEO by NEO basis.

**Accessing raw AXP209 information using i2c**

I've written a python script to do this access, "RW\_AXP209-2.py" (also updated to "...-3.py") to directly read the AXP209 registers. After enabling ssh (see above) log into the ConnectBox software (root, connectbox) and load the smbus module with:

sudo apt install python-smbus

Then load the script to the NEO (see above... file transfers) and run the script with

python RW\_AXP209-3.py

**On tracing through the neo\_batterylevelshutdown.py code**

Edwin supplied that the neo\_batterylevelshutdown code on the neo can be stopped and restarted with:

systemctl stop neo-battery-shutdown and

systemctl start neo-battery-shutdown

After starting, the internal messages generated by the code can be monitored by:

journalctl --follow --unit=neo-battery-shutdown // break out with Ctrl-C

This facility proved very useful in tracing the "no display" issues introduced after coderelease 0914. Tracing in release 1119 was able to resolve the issue was from a missing C module. To install this module in the 1119 release you can run:

apt-get install libfreetype6

**Other commands from @edwin**:

dmesg // reads debug messages created by system (particularly for driver issues)

cat /sys/class/net/wlan0/device/uevent // info on the wifi modules (empty if not running)

iw list // more info on wifi module

**More...**

ifconfig

iwconfig

iwconfig; dmesg | tail -20; tail -20 /var/log/daemon.log; cat /etc/hostapd/hostapd.conf

cat /etc/hostapd/hostapd.conf

systemctl cat hostapd.service

ps -ef | grep hostapd; hostapd\_cli status

/var/log/daemon.log //(the place where some services write their logs)

last // shows previous status (from last "shutdown"

**python for I2C without smbus library download...**

/usr/local/connectbox/battery\_tool\_venv/bin/python

**If wifi SSID not showing... try...**

hostapd\_cli status

and send to edwin

Installing i2c-tools

sudo apt-get install i2c-tools

i2cdetect -r 0

Location of neo-battery code modules (on NEO):

/usr/local/connectbox/battery\_tool\_venv/lib/python3.5/site-packages/neo\_batterylevelshutdown/

Also, one level up is axp209.py

In neo\_batterylevelshutdown, find hats.py. At about line 47 we have the shutdown function where we could write the command to cause charge current to be back to "normal" (about 1 A). Register 33H [3:0] controls charge current.

(Ichg = (300 + (Bit 3-0) \* 100 mA). SO 1A limit is 0x7; and a 400mA limit would be 0x1).

Also in same directory, in hats.py, we find the classes for the various hat types.

command of interest:

self.axp.bus.write\_byte\_data(AXP209\_ADDRESS, 0x43, 0x41)

self.ax[/bus/read\_byte\_data(AXP209\_ADDRESS, 0x32)

In hats.py, we also find function \_\_init\_\_ (self, displayClass): (line 159) where all the AXP209 setup is done... this is where we would add a write to Reg 33H to set the charge current at something low (like 400 mA)

Read of Reg 33H (normal unit) = 0xC9 (1.2A)

in vi, turn on line numbers with:

:set number (turn off with :set nonumber)