Introduction to Raspberry Pi.

1. Raspberry PI runs an operating system version of linux – this one is a small distribution of Debian called Raspbian. We have installed the light version known as “Jessie”.

2. We will connect to the RasPi using a **headless pi** configuration. If you need to hunt for solutions or config settings, use “headless pi” as keywords when Googling. The other option is to treat the Pi like a small computer and connect keyboard, mouse and monitor. We will not take that approach, but instead we will log into the Pi in the same way we logged into the Oscar HPC (minus the two-step verification).
3. Raspbian lite allows all the tools of linux, but without the graphical interface. More info found here: https://www.raspberrypi.org/downloads/raspbian/

Steps to configure and install (you’ll begin with Step 4).

1. **Burn** disk image onto sd card using etcher.dmg or similar (already done).
2. **Add** a file called ssh to the boot directory. This allows us to connect to the pi by Secure Shell (already done).
3. **Edit** cmdline.txt inside the boot folder of the disk image. Add ip=169.254.200.127 with one whitespace directly after rootwait. Don’t add any tabs or returns to the line or it will mess up the formatting (already done).
4. **Connect** your Pi to the Ethernet port on your computer using an Ethernet cable. If your computer does not have an ethernet port, we have USB to ethernet adapters available.
5. **Insert** the micro SD into the slot on the under-side of the Pi.
6. **Give** power to the RasPi by connecting the micro USB to your computer or external power supply.
7. **Confirm** that your computer can ‘see’ the headless Pi on the network:
   a. MAC: Open a terminal. Use command:
   $ ping 169.254.200.127 # you should see a response that looks like this:

```
64 bytes from 169.254.200.127: icmp_seq=5 ttl=59 time=20.682 ms
64 bytes from 169.254.200.127: icmp_seq=6 ttl=59 time=19.996 ms
```

   $ ping raspberrypi.local # is also an option, but its preferred to use the IP that we already set.

   b. WINDOWS: From the Windows menu, type CMD in the search menu to bring up a terminal. Use the ping command to confirm that the Pi and your computer can ‘see’ each other on the network.
   $ ping 169.254.200.127 # you should see a response with data.

```
64 bytes from 169.254.200.127: icmp_seq=5 ttl=59 time=20.682 ms
64 bytes from 169.254.200.127: icmp_seq=6 ttl=59 time=19.996 ms
```

If successful, then proceed to login using SSH.

8. **Use Secure Shell** to make an encrypted Telnet connection to the Pi. You can use the same procedure as with the Oscar HPC connection. On Windows download Putty or other encrypted telnet program: https://www.putty.org/

$ ssh pi@169.254.200.127 (or something like this). Pi is the username and password is raspberry.

If this connection is successful, then you can proceed to testing out the file system. Move around the directories with **cd**. Check out where you are using **pwd**. Make a new text file, using **nano**.

Notes about Linux: If you haven't already, please be sure to follow this tutorial on Unix/Linux shell commands. Also, check out Unix shell commands to better understand how to use the command line tools. https://maker.pro/linux/tutorial/basic-linux-commands-for-beginners. You will likely need to use the Unix commands **sudo**, **cd**, **rm**, **mkdir**, **ssh**, **sftp**, and **nano.** Those who are more interested in Unix should explore 'Unix shell scripting' as a topic.

In-class: Use RasPi as computer to receive and record temperature data collected by the Arduino.

1. Edit serialpie.py to create a Python script to capture resistance from the arduino and record it to a csv file on the Raspi. You will follow a very similar procedure to the one you used for the therm_plotter lab, and you can recycle much of that code.
2. Complete a first draft of serialpie.py and upload to the Raspi using sftp, similar to commands used for the HPC exercise.
3. Use the serial cable to connect the Arduino to the Pi. Once the Pi is powered up, this should also power the Arduino.
4. Connect to the raspberry pi with ssh. Execute serialpie.py on the Raspberry Pi, read inputs from the Serial port and print those to the screen.
5. Use the Python csv writer to create a data file and write the datetime, and temperature data that comes from the Arduino on a file from the Pi, just as you did for the therm_plotter.ipynb.

What to turn in?

Go through the exercises above. Turn in your serialpie.py script showing your Pi is reading from the Arduino and printing the contents to the screen, as well as saving those contents to a csv file.