RASPBERRY PI2 DEMO

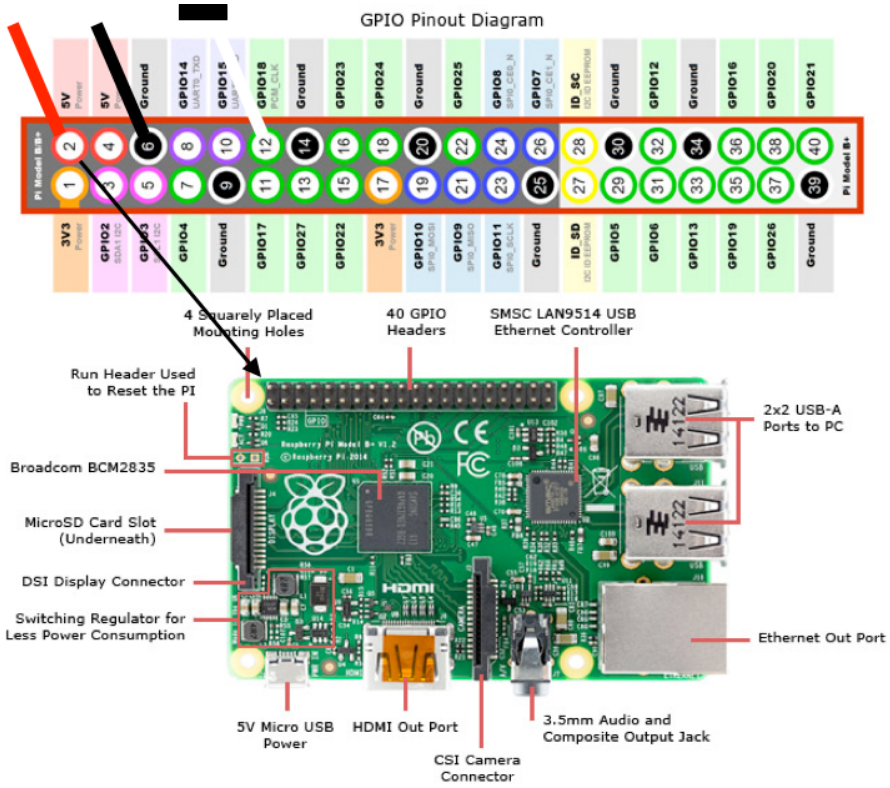
NeoPixels

In this demonstration we will learn how to:
1) use a Raspberry Pi remotely
2) connect simple hardware to the Pi
3) Write Python programs to make the hardware do something

This specific demo will make use of a “NeoPixel Strip”. Which is a set of 8 multi-colored programmable LEDs (light emitting diodes). You will learn how to connect the strip, and how to program the color of each pixel.

Some of the things we are doing in this demo only work because we are using such a small NeoPixel strip. If you use a larger number of NeoPixels, you need to have a different strategy for giving it enough power, and for preventing damage to the strip. If you purchase a RasPi and NeoPixels yourself, please don’t just follow this demo, but check the official page:

https://learn.adafruit.com/neopixels-on-raspberry-pi/overview



Step 1 - Plugging in the hardware

The first step is to attach our NeoPixel strip to the RasPi. The strip communicates using a very simple 1-wire interface. There are 2 more wires to provide power. Since our strips are short, we will power them directly from the RasPi. Longer strips may draw too much power to use this method.

1) **Note that you can destroy the NeoPixel strip and/or the RasPi by plugging the wires in the wrong place.** Please be careful, and ask if you don’t understand. That said, it isn’t rocket science, it is pushing 3 wire sockets onto 3 wires.

2) While it is ok for the RasPi to be turned on/plugged in while you do this, off is always safer.

Note that some of the NeoPixel sticks have a 6-pin connector on them. This makes them even easier to plug in, but be VERY sure that the red wire goes on to the correct pin.

3) You must plug the wires in a specific sequence if the power is on. Plugging them in in the wrong order can damage the NeoPixels.
4) First, plug the black (or blue) wire (ground) into Pin #6 (see the diagram below)
5) Second, plug the red wire (+5v power) into Pin #2
6) Third, plug the white (or grey) wire (comm) into Pin #12 (the 6th pin from the left on the top row)
7) That's it. Everything is connected. Normally nothing will happen at this point. Sometimes you will see a random pixel or two change color. If so, ignore it, we will deal with it soon

Step 2 - Talking to the Raspberry Pi

Normally when using a RasPi, you would connect its HDMI output port to a monitor, and plug in a USB keyboard and mouse. Since I don't have that much equipment available, we will communicate with our RasPi's over the network. Each of the Raspberry Pi units in the lecture hall today have a WiFi adaptor on them, and they have been programmed to connect to the "IP" WiFi network we used in the networking demo.

For this to work, you need to have an SSH client installed on your computer. On Mac/Linux this should be part of the operating system. On Windows, I suggest you install a free program called "Putty".

1) RasPis are powered via USB. A standard USB port on a laptop or hub can deliver ~500 mA of power. The RasPi draws about 300 mA and if you have all of the pixels on at full brightness, they draw ~160 mA. So, it *should* be fine to power your RasPi from your laptop (I do it all the time), and almost all laptops have over-current protection so *if* you draw too much power it will just disable the USB port until you reboot. That said, if you are worried, I have a bunch of USB power supplies at the front, and you are welcome to plug into one of these instead of your laptop if you like. Since you are talking to the RasPi over WiFi, it doesn't matter if it's across the room. USB is not being used for communications, just power.
2) Each of the Raspberry Pis we are using has a number on it. Make a note of this number, particularly if you decide to plug it in at the front of the room instead of using your laptop.
3) Switch your laptop from the BCM network to the "IP5" or "IP" WiFi networks. If you see a warning message about the WAN being disconnected in this step, just ignore it and close the window.
4) The address of each RasPi will be 192.168.5.# where # is the number on the unit. Do not accidentally connect to someone else's Pi. Now:
    1) (mac/linux) type: ssh pi@192.168.5.#
    2) (windows) launch Putty and connect to 192.168.5.# with the username "pi"
5) The password is "hardware2"
6) You should now have a linux command prompt, like:

pi@raspberrypi:~ $

7) If you don't get this, please ask for help. Don't continue with the next step until you have this prompt.

Step 3 - Programming the NeoPixel

Now we'd like to make our pixels do something. Luckily, someone has made a very nice Python library available for this purpose. Since it needs to access the hardware directly on the RasPi, it will need to run with administrative (root) privileges, but that's ok. The account you are using has permission to do this using the 'sudo' command.

Most of these hardware RasPi libraries still work with Python 2 rather than Python 3. The differences are pretty minor, however, so you should largely be able to do all of the normal things you are used to, even though you are using a Python 2.7 interpreter.

We'll start with a simple demo. Type the following in your SSH window on the RasPi:

1) sudo ipython

This should give you the familiar iPython prompt. The 'sudo' command gives you administrative privileges, which is required to access the NeoPixels directly.

2) Now type the following commands (you can skip the comments) into Python. Note that ipython supports autocompletion, so, for example, on the first line, you could just type strip=Ada then hit <tab> and it will type as much of the rest for you as it can:

```
from neopixel import *
# 8 is the number of pixels, 200 is the maximum brightness, you can use 255 if you prefer
# the other numbers are all fixed
strip=Adafruit_Neopixel(8,18,800000,5,False,200)

# initialize the strip
strip.begin()
strip.setPixelColorRGB(0,128,0,0)     # changes the color, but doesn't take effect until show()
strip.setPixelColorRGB(1,0,128,0)
strip.setPixelColorRGB(2,0,0,128)
strip.show()
```

3) You should now see 3 of the pixels lit up, one red, one green and one blue. An easier way to program the values of many pixels is with:

```
# 'x' will become a list-like object with 8 values, representing the color of each pixel
# as above, after changing the values, you must show() to see the result
x=strip.getPixels()
x[3]=0x808080          # pixels colors are a 3 byte number, 0x means hexadecimal
strip.show()
```

Step 4 - Writing your own program

You are using the RasPi via a terminal session, so you cannot open graphical editor windows to write programs. However, there are some terminal based text editors you can use via SSH which still work pretty well. The easiest one to use is called 'nano'. If you type:

```
nano myprogram.py
```

It will open an editor where you can edit your program. Since this isn't mouse-based, you have to use control keys to exit the editor and perform other operations. The basic control keys are shown at the bottom of the editor window. If you see "^X" for example, this means hold down the control key and press X.

Try writing a program to do something interesting with the 8 pixels you have. Remember to run your program as:

```
sudo python myprogram.py
```

Step 5 - Shutting down

When you are done, unplug the USB cable powering the RasPi FIRST, then unplug the NeoPixels.