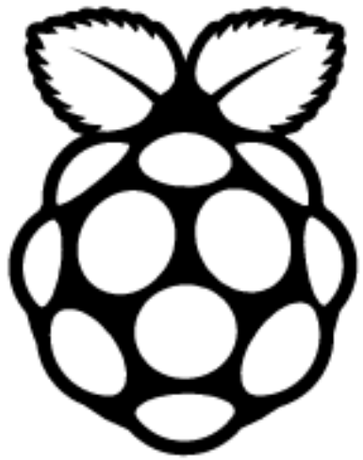
Raspberry Pi 1

Getting started



Owen Mundy | Fall 2013

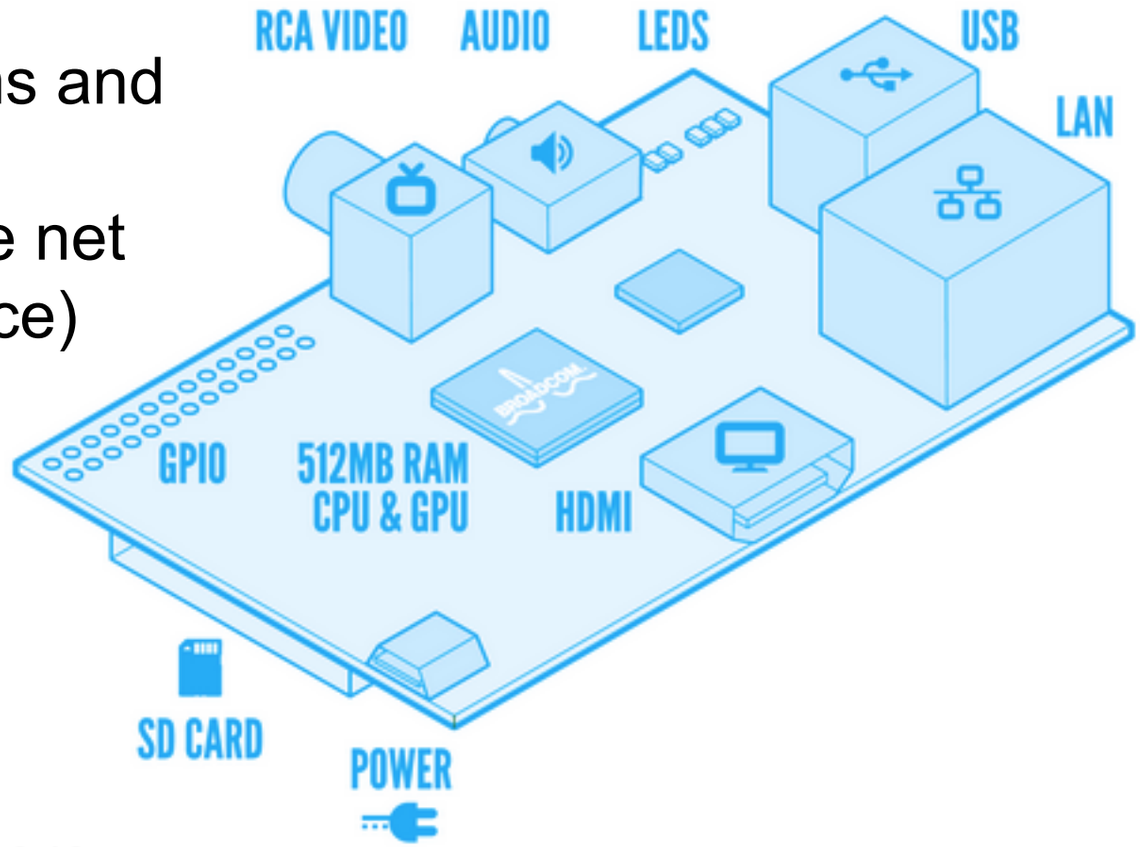
Overview

- Introduction to the Raspberry Pi
- Installing the OS onto an SD Card
- Connecting to the Pi for the first time over SSH
- First time configuration
- Connecting to the Pi over SFTP
- Moving around and executing files with SSH
- Write your first Python script

What is the Raspberry Pi?

- A small computer that runs on the Linux Operating System. It contains the following ports (moving clockwise in the image below):

  - RCA video - Analog video out
  - 1/8 in. (3.5mm) Stereo audio - Analog audio out
  - On-board LEDs for monitoring status
  - 2-USB ports for mice, keyboard, webcams and other peripherals
  - LAN or ethernet port for connecting to the net
  - HDMI (High-Definition Multimedia Interface) video output for a TV or monitor (digital signal w/sound)
  - On-board micro-USB power
  - Removable (and upgradable!) SD card for the OS and files
  - GPIO (General Purpose Input/Output) pins for connecting to sensors and actuators
  - 512MB RAM for the CPU and GPU

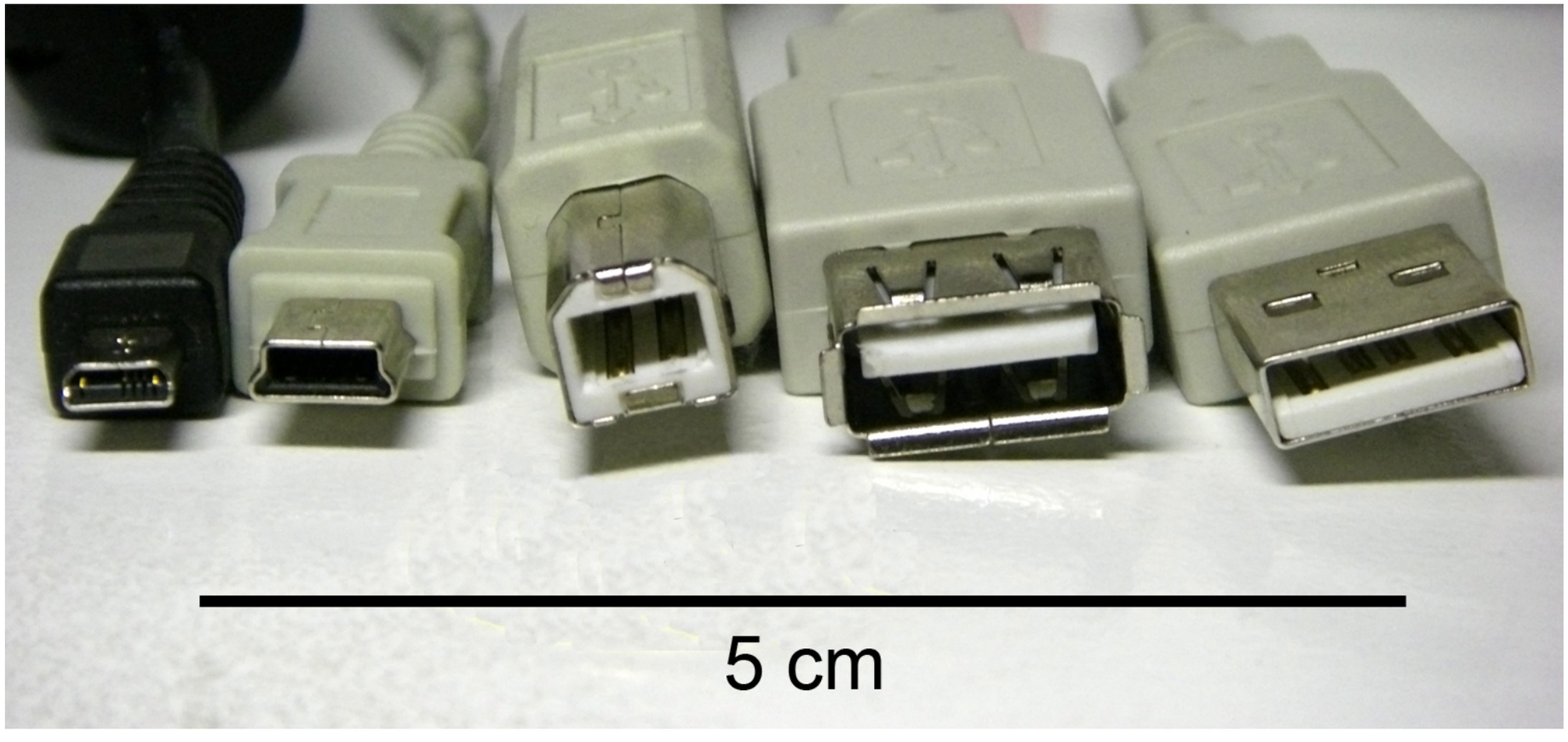


Source: Monk, Simon. Programming the Raspberry Pi: Getting Started with Python

Powering your Pi

- The Raspberry Pi requires proper power supply to function properly:

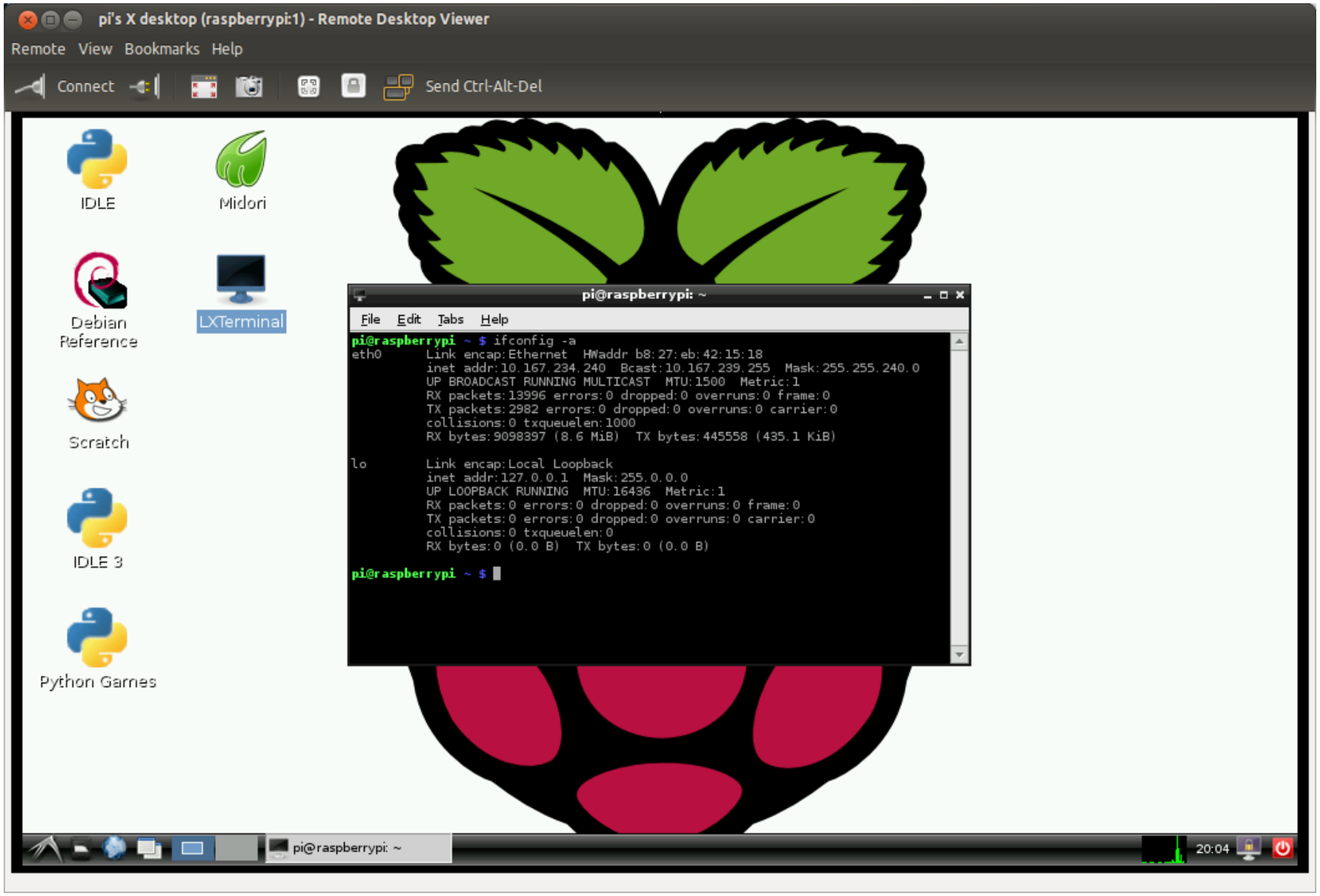
  - Voltage (V): **5V DC**
  - Current (mA): **ideal: 1000mA**, range: >= 700 mA, < than 2000 mA
    *USB from your computer only has 500 mA, so will not work
  - Interface: USB micro (not mini!)
  - See also: Testing Raspberry Pi's Power Supply



Five common USB connectors (left to right: micro USB, male Mini USB (5-pin) B-type, male B-type, female A-type, male A-type)

The operating system (OS)

- The Raspberry Pi can run various *flavors* of the Linux OS.
- Unlike Arduino, the Pi has a Graphical User Interface (GUI) and a desktop.



Source: Monk, Simon. Programming the Raspberry Pi: Getting Started with Python

Operating systems

- The Raspberry Pi foundation lists 6 different operating systems on their download page (as of Sept 2013)

Raspbian: A Debian wheezy port, optimized for the Raspberry Pi
- Great for beginners. I've used this for many projects.
- Dependable package manager
- A NOOB package is provided to make installing the OS "simpler" (but we will be doing it the regular way).

RISC OS: UK version claiming very fast and compact system
- I have not tested this
- Don't forget to change keyboard to U.S.

RaspBMC: An XBMC media center distribution for Raspberry Pi
- I tested this and found poor support for a variety of keyboards so I decided to wait for some upgrades to happen before I tried again.

Source: Monk, Simon. Programming the Raspberry Pi: Getting Started with Python

Installing the OS onto an SD Card

The following steps require sudo (temporary superuser privileges) access so this will need to be performed on a machine where you have admin access. You also need an SD card reader.

1. Download and unzip the Raspbian image

2. Plug a >= 4GB SD card in your Mac. (We will be erasing all content on that disk so make sure there is nothing on there you want to keep.)

3. In Terminal, run the following command to determine which disk is the SD card
   diskutil list

4. Unmount the partitions by typing (replacing disk**2**s1 and disk**2**s2 with your **#**)
   diskutil unmount disk**2**s1
   diskutil unmount disk**2**s2 (may not exist on a new SD card)

Also see: RPi Easy SD Card Setup

Installing the OS onto an SD Card

5. Copy image to SD card (replacing **red** text) reference (note number at rdisk)
   sudo dd bs=1m if=./**2013-09-25-wheezy-raspbian.img** of=/dev/rdisk**1**

This part could take a while... Last time it was 79 minutes...

```
Owens-Macbook-Pro:Downloads owenmundy$ sudo dd bs=1m if=./2013-09-10-wheezy-raspbian.img of=/dev/rdisk4
Password:
3781+1 records in
3781+1 records out
3965190144 bytes transferred in 954.397101 secs (4154654 bytes/sec)
Owens-Macbook-Pro:Downloads owenmundy$
```

6. Eject the SD card (where disk2 is your disk)
   diskutil eject disk**2**

7. Put the SD card in your PI and start it up!

Also see: RPi Easy SD Card Setup

Connecting to the Pi for the first time

Before we can connect we have to make sure Avahi is running on the Pi. Plug it into a display and keyboard/mouse and enter the default username/password:

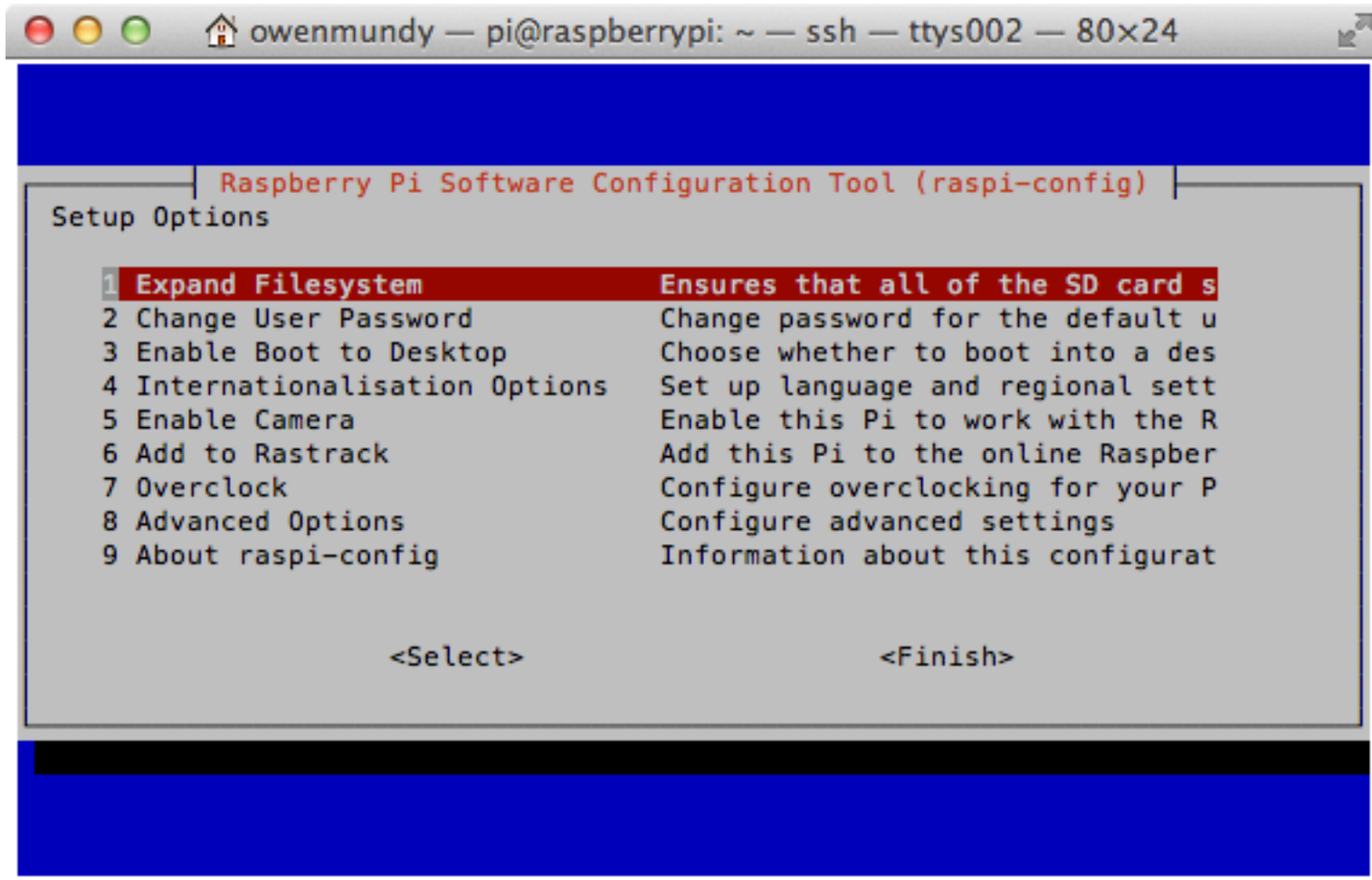
user: pi
pass: raspberry

Configuring Raspbian

Now let's configure Raspbian using the Raspberry Pi config "Boot menu." Type:

sudo raspi-config

User the cursor keys to highlight the correct options and type enter.



Configuring Raspbian

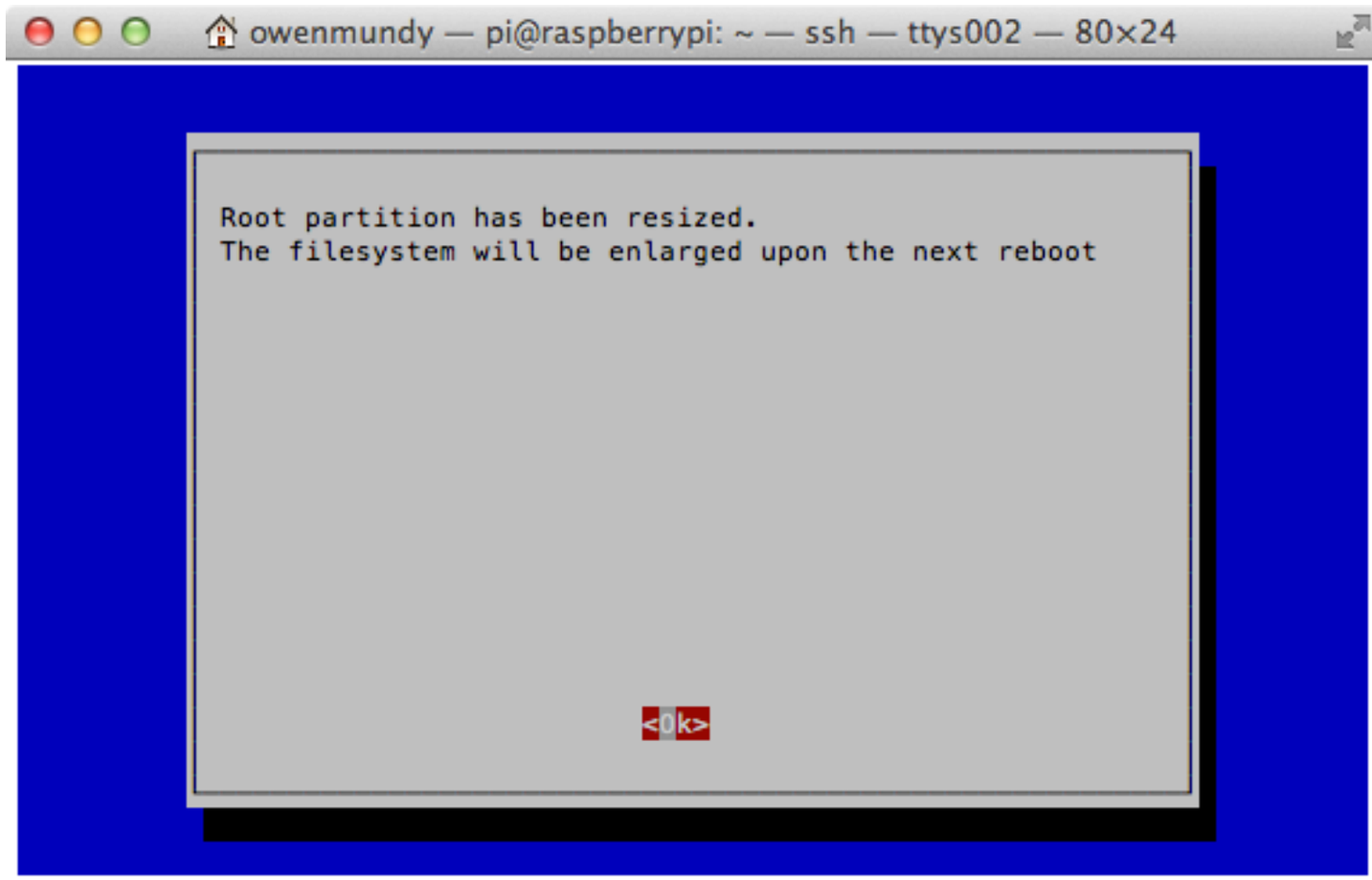
1. Expand Filesystem = yes

This expands your root filesystem to use the entire SD card.

More info: Manually resizing the SD card partitions

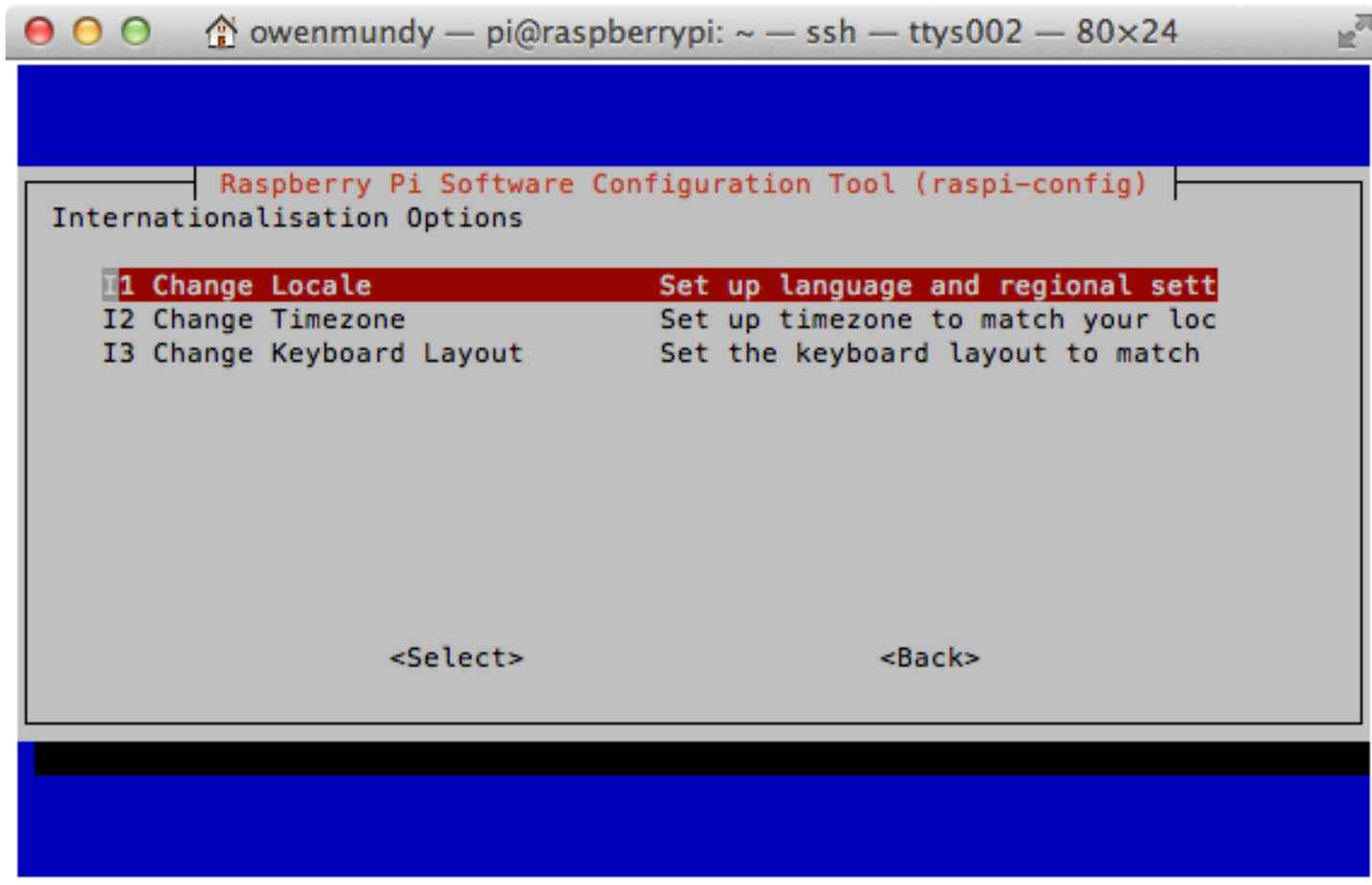
2. Change User Password = no
   - While generally a good idea we'll skip this step for now

3. Enable Boot to Desktop = no
   - Automatically boots to the desktop on startup. We won't enable this for now.



Configuring Raspbian: Internationalis|zation

4. Internationalization Options
   - Change Locale = en_US.UTF-8 (the default is "en_GB.UTF-8")
   - Change Timezone = America > New_York



Use cursor keys and type space to select correct option.

5. Enable Camera = yes

6. Registration = no

7. Overclock = overclocking may reduce the lifetime of your Raspberry Pi
   - Overclock a Raspberry Pi without Voiding Your Warranty

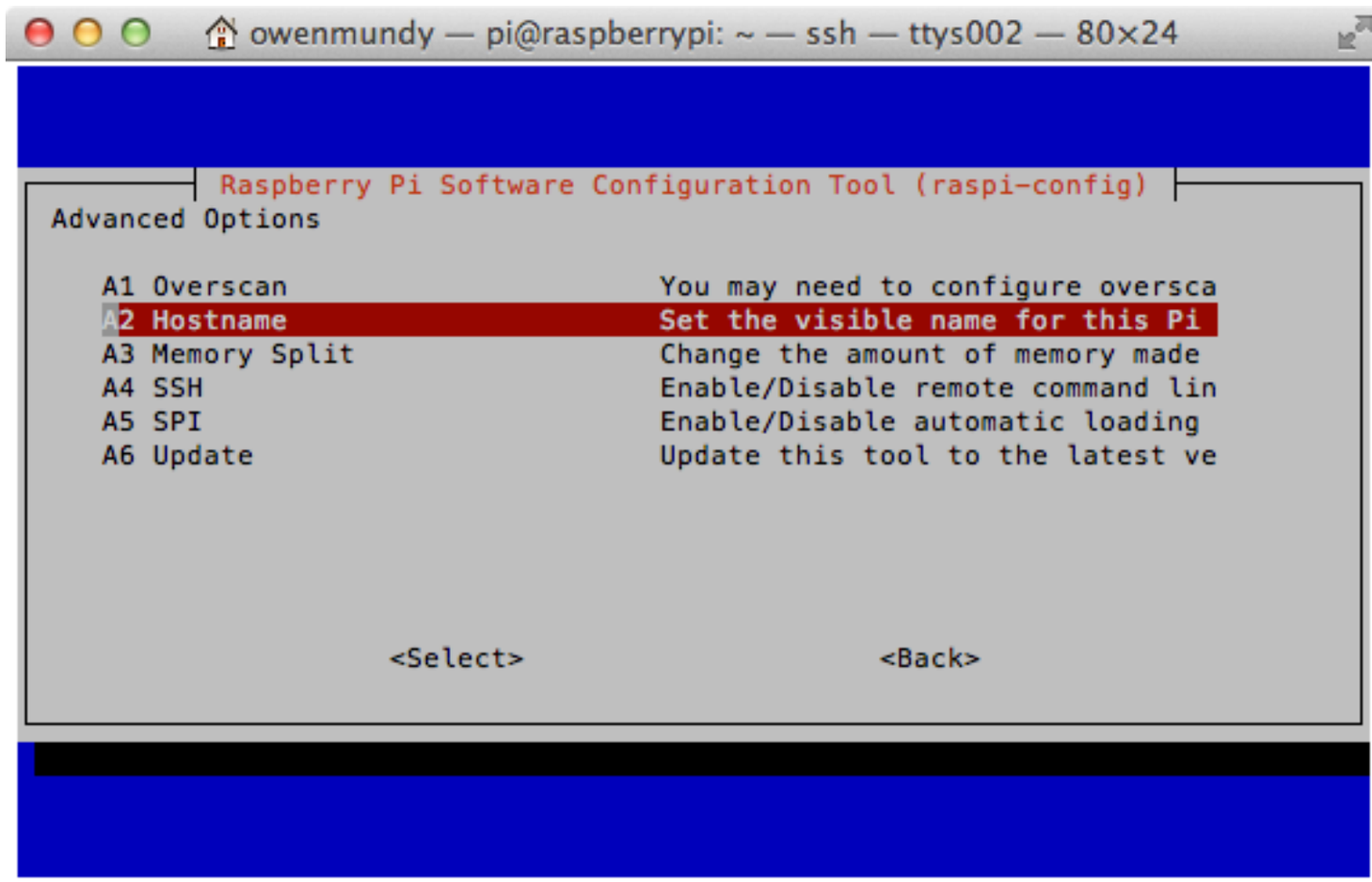
Configuring Raspbian: Internationalis|zation

8. Advanced Options
   - Hostname = change from raspberrypi to something you know.
     For example, I'll now login with pi@robomonitor1.local

   - SSH = yes Enable/Disable remote command line login

   - Update = yes

Restart for changes to take effect
sudo reboot



Connecting to the Pi for the first time

Before we can connect we have to make sure Avahi is running on the Pi.

To launch the GUI:

startx

Then, following these instructions: http://elinux.org/RPi_Advanced_Setup

1. Install avahi with the following commands on the Pi:

   sudo apt-get install avahi-daemon

2. Now this command

   sudo insserv avahi-daemon

3. Create a configfile for Avahi

   sudo pico /etc/avahi/services/multiple.service

Connecting to the Pi for the first time

...Continuing from these instructions: http://elinux.org/RPi_Advanced_Setup

4. Paste this into the config file and save w/ CTL+X

```
<?xml version="1.0" standalone='no'?>
<!DOCTYPE service-group SYSTEM "avahi-service.dtd">
<service-group>
        <name replace-wildcards="yes">%h</name>
        <service>
                <type>_device-info._tcp</type>
                <port>0</port>
                <txt-record>model=RackMac</txt-record>
        </service>
        <service>
                <type>_ssh._tcp</type>
                <port>22</port>
        </service>
</service-group>
```

Connecting to the Pi for the first time

...Continuing from these instructions: http://elinux.org/RPi_Advanced_Setup

5. Apply the new configuration with:
    sudo /etc/init.d/avahi-daemon restart

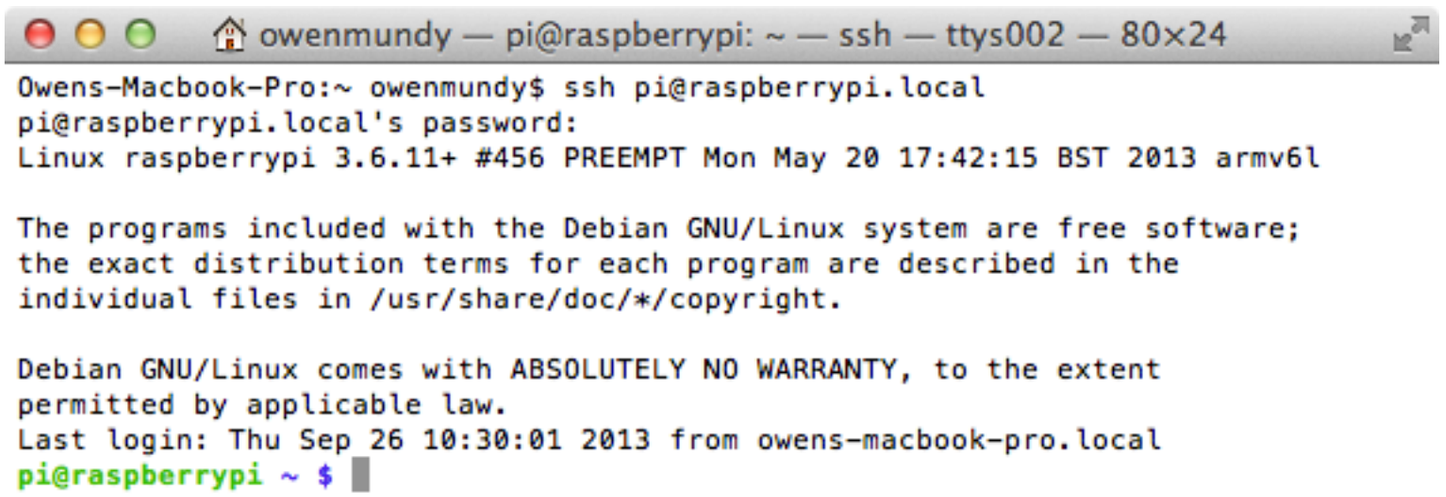
Connecting to the Pi for the first time

Now we can SSH to the Pi for the first time *with your new hostname*.

In Terminal:

ssh pi@raspberrypi.local

Type yes to agree and then enter the password. Hello Pi!



Looking around

By default we land in the home directory, denoted by a tilde “~”.

We can always return here by typing:

cd ~

We can confirm our current directory

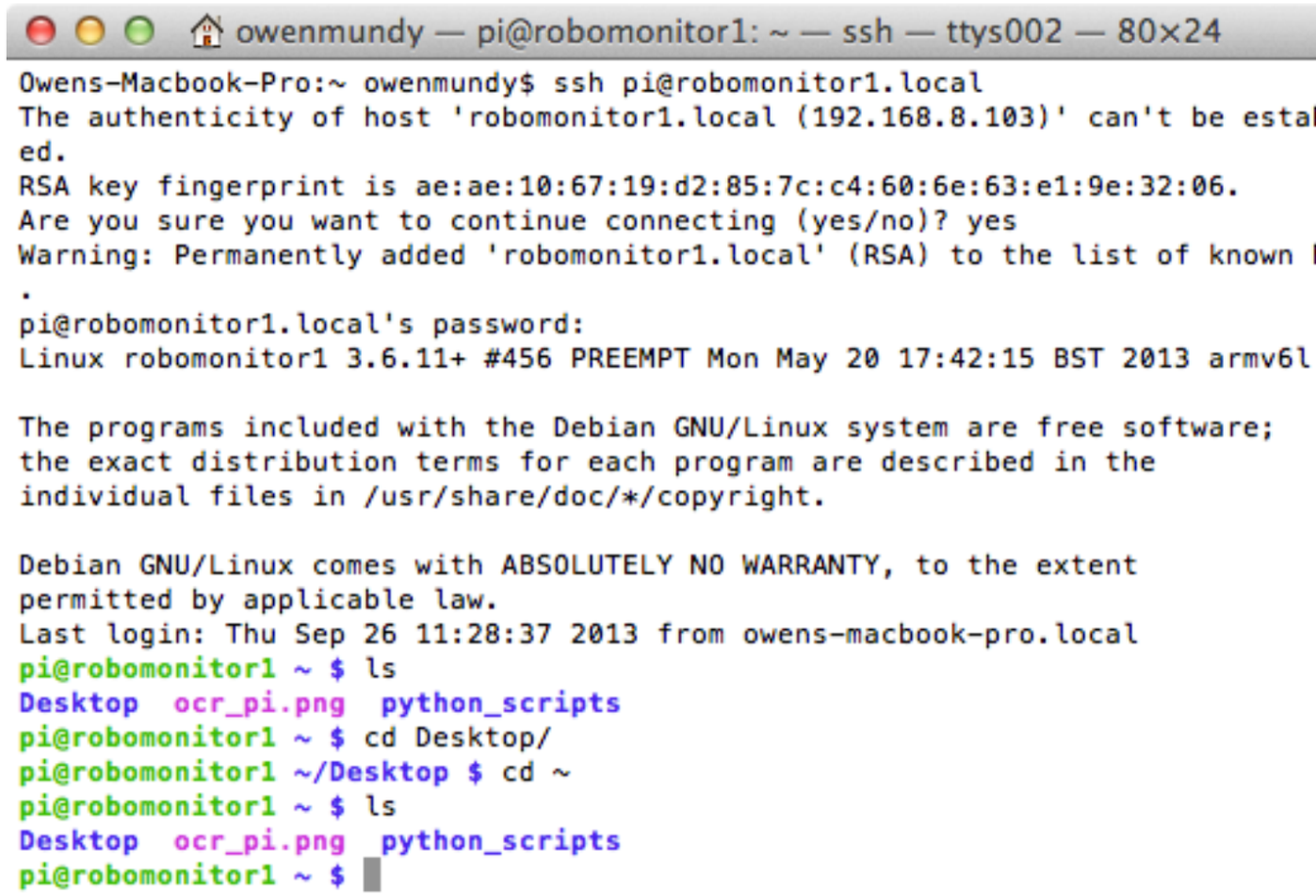
pwd

Let’s have a look around:

ls

Or look in another directory

ls Desktop



Connect over SFTP

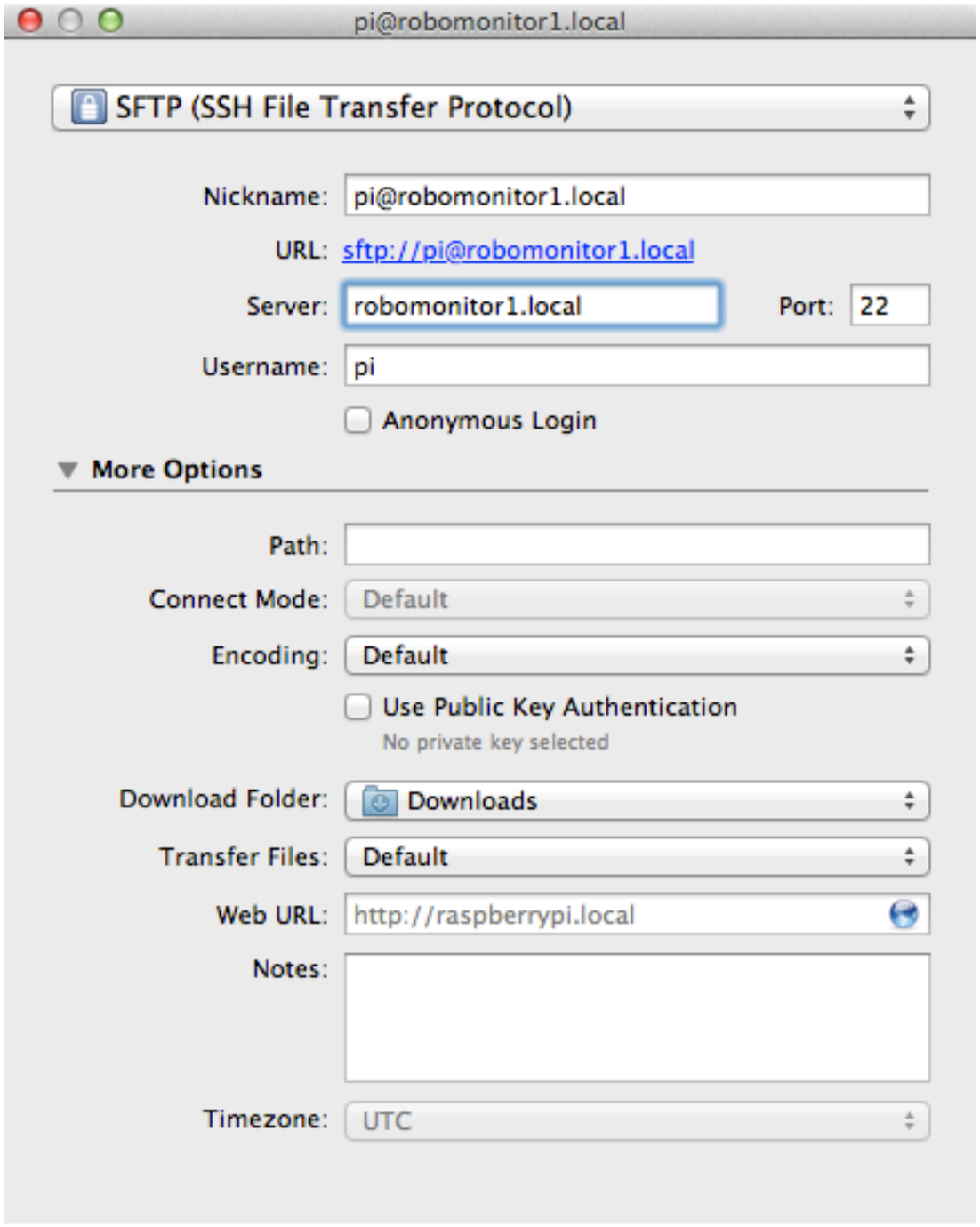
Here's a cool trick for writing scripts.

You can connect via SFTP and open files in your favorite editor.

Launch an FTP application (like Cyberduck) and make a new bookmark.

Protocol: SFTP (SSH File Transfer Protocol)
Nickname: pi@robomonitor1.local
Server: robomonitor1.local
Port: 22
Username: pi

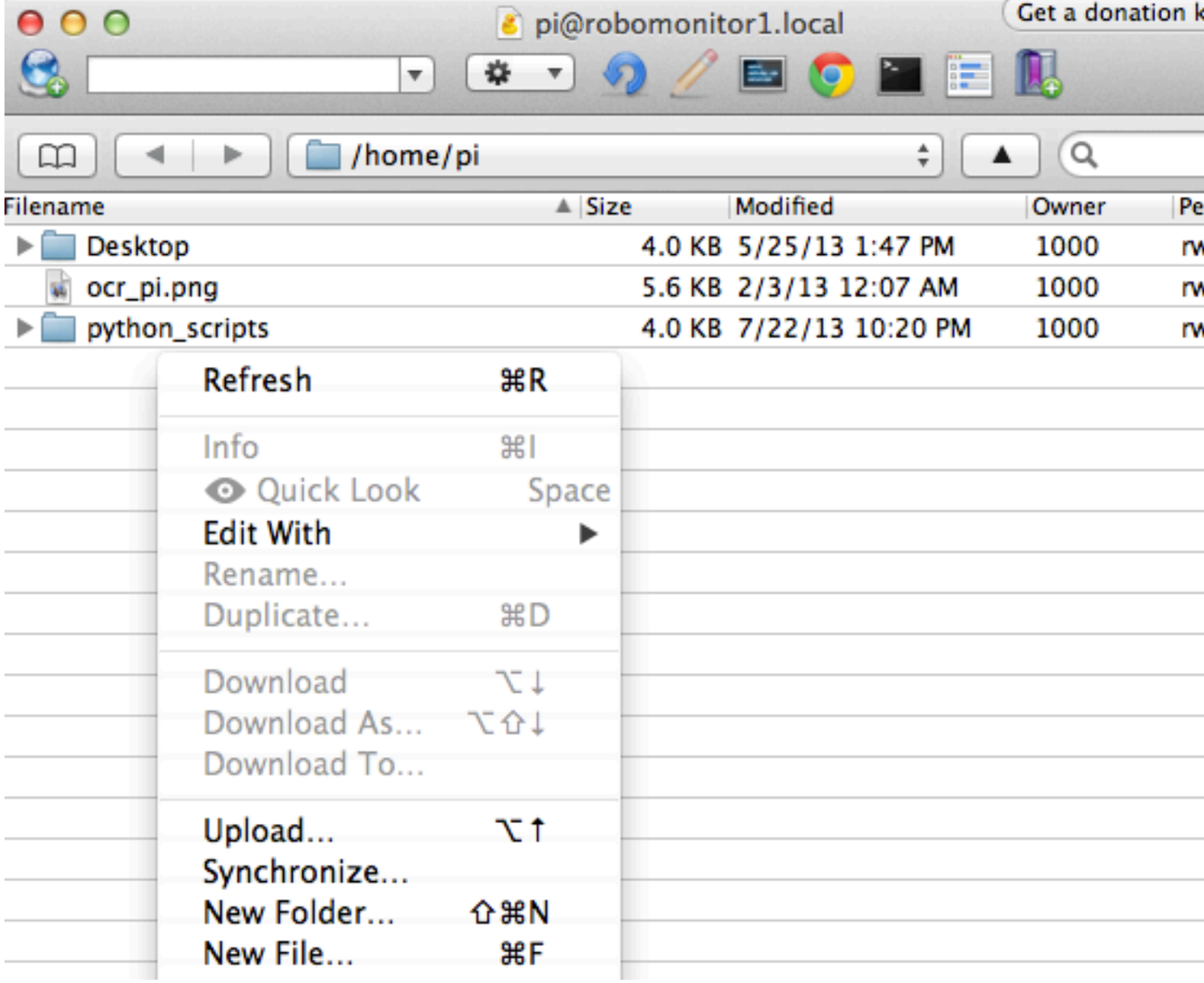
Close bookmark, double click, and enter password!

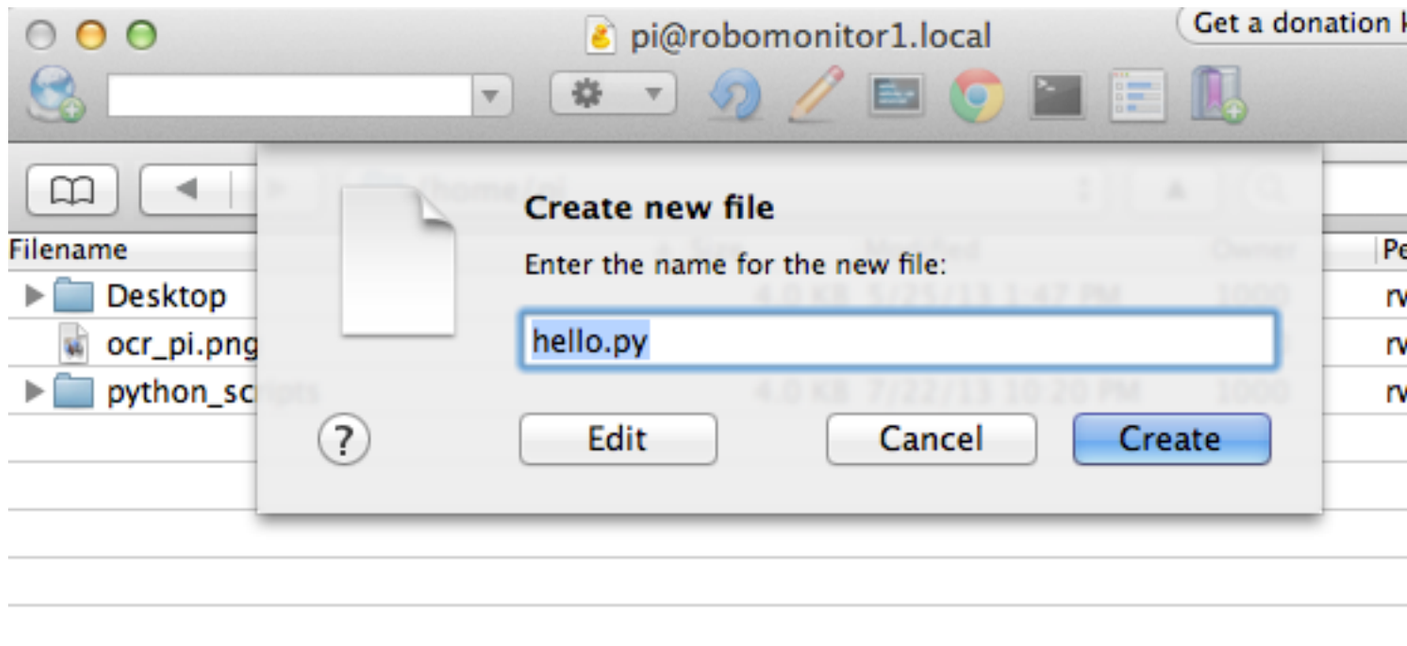


Make a new file

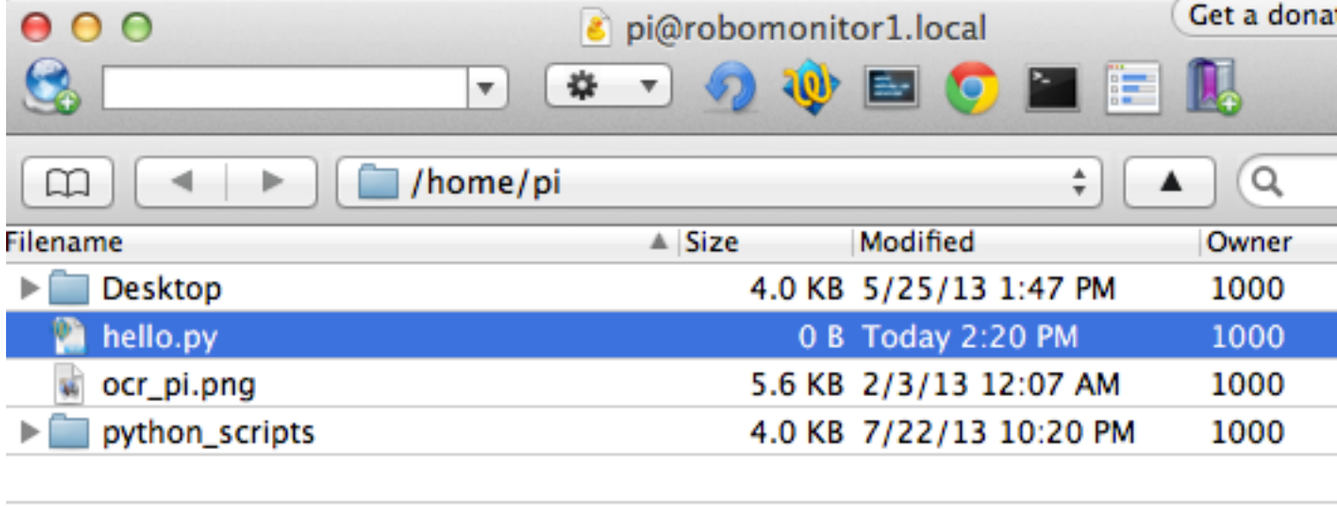
Once you're connected then CTL +click and create a new file.

Use something simple like: "hello.py" (yes, we are about to write our first Python script :-)





Edit a new file



Now highlight the file
and press Command + K

This should open your preferred text editor. The can be changed in Cyberduck > Preferences.

Start by typing a shebang (#!) and path to interpreter at the top of the file

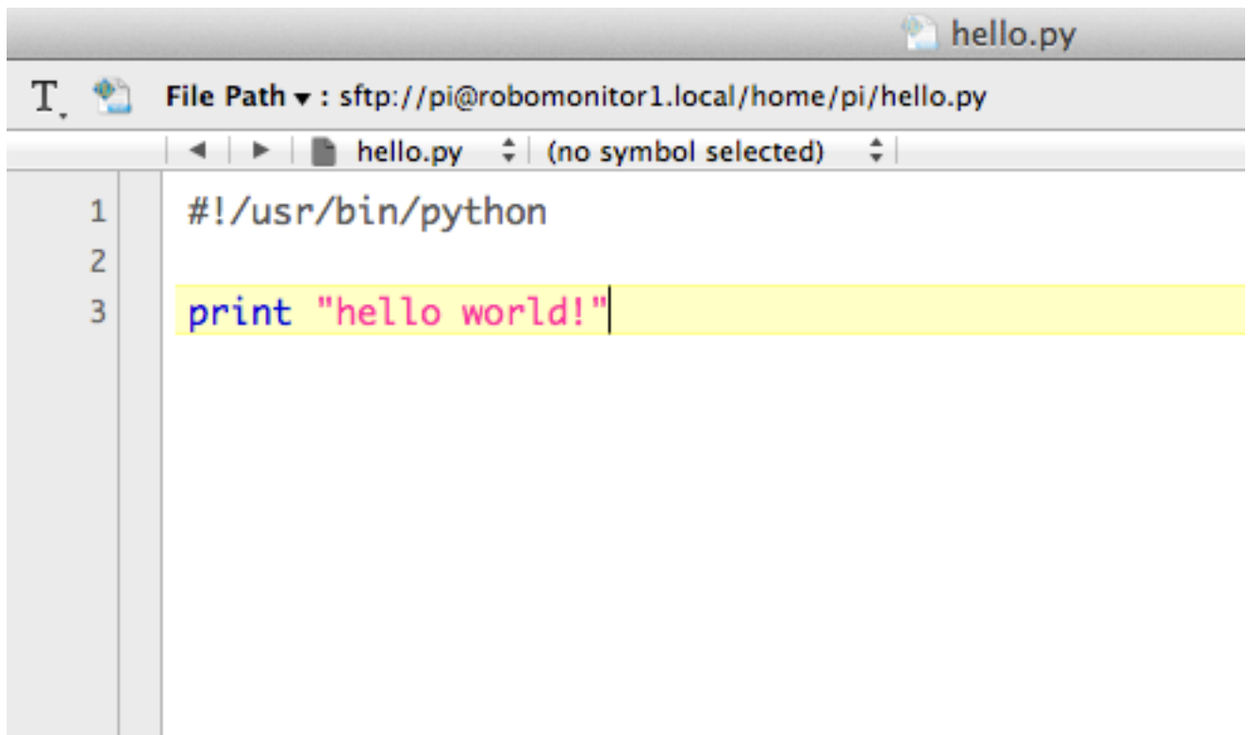
```
#!/usr/bin/python
```

This tells the system that this file is a script and attempts to execute it using specified path/to/specified/interpreter. It also allows us to make it an executable file.

Now type:

```
print “hello world!”
```

And save the file and it should be uploaded to the Pi automagically.

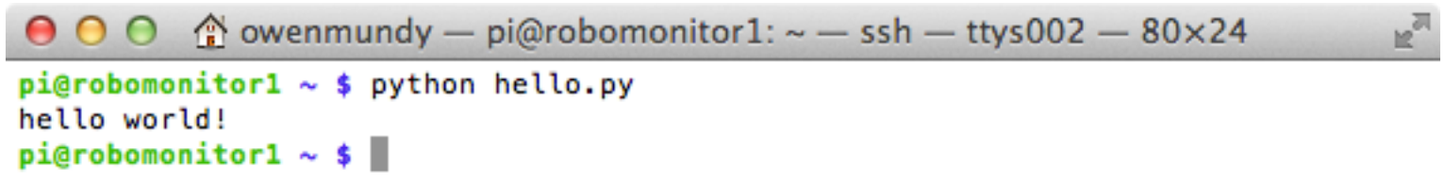


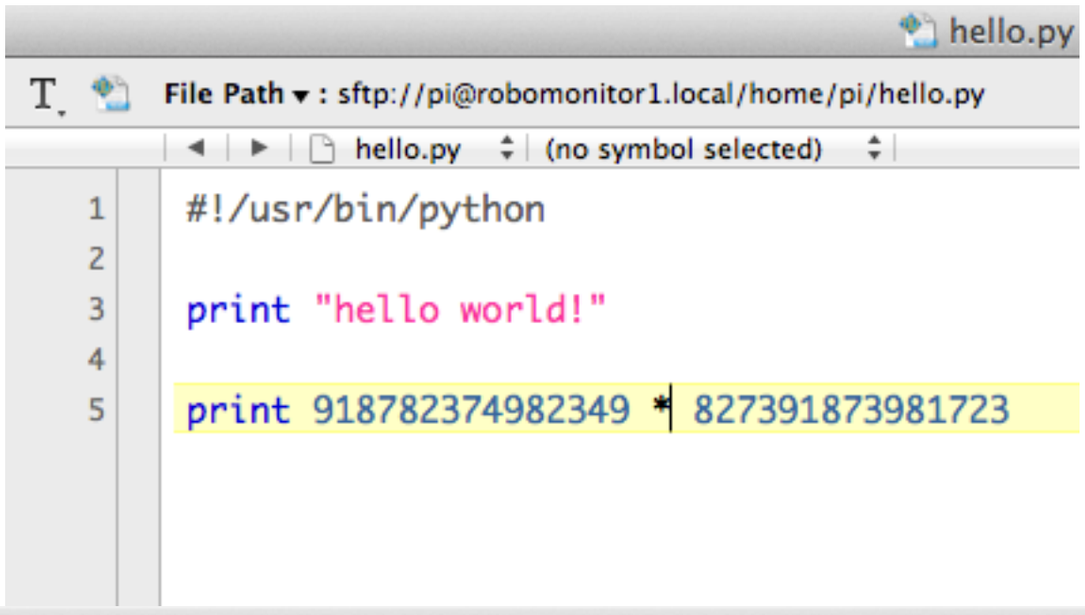
Edit a new file

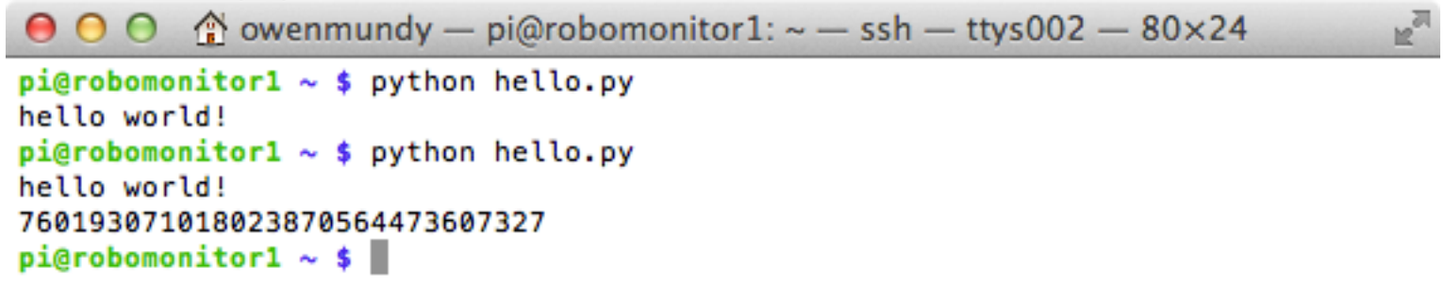
Now in Terminal type:
python hello.py

This tells the Python interpreter to execute this file.

Edit the file and see what else you can do! Just press up arrow in Terminal to show most recent commands and press enter to execute the file again.

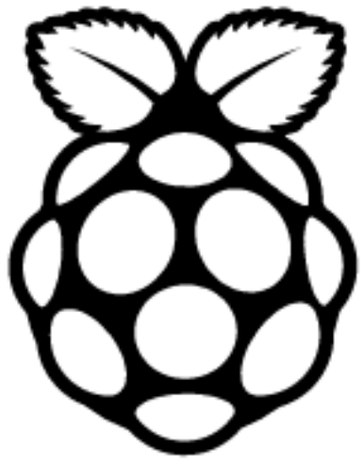






Raspberry Pi 2

Package managers, python



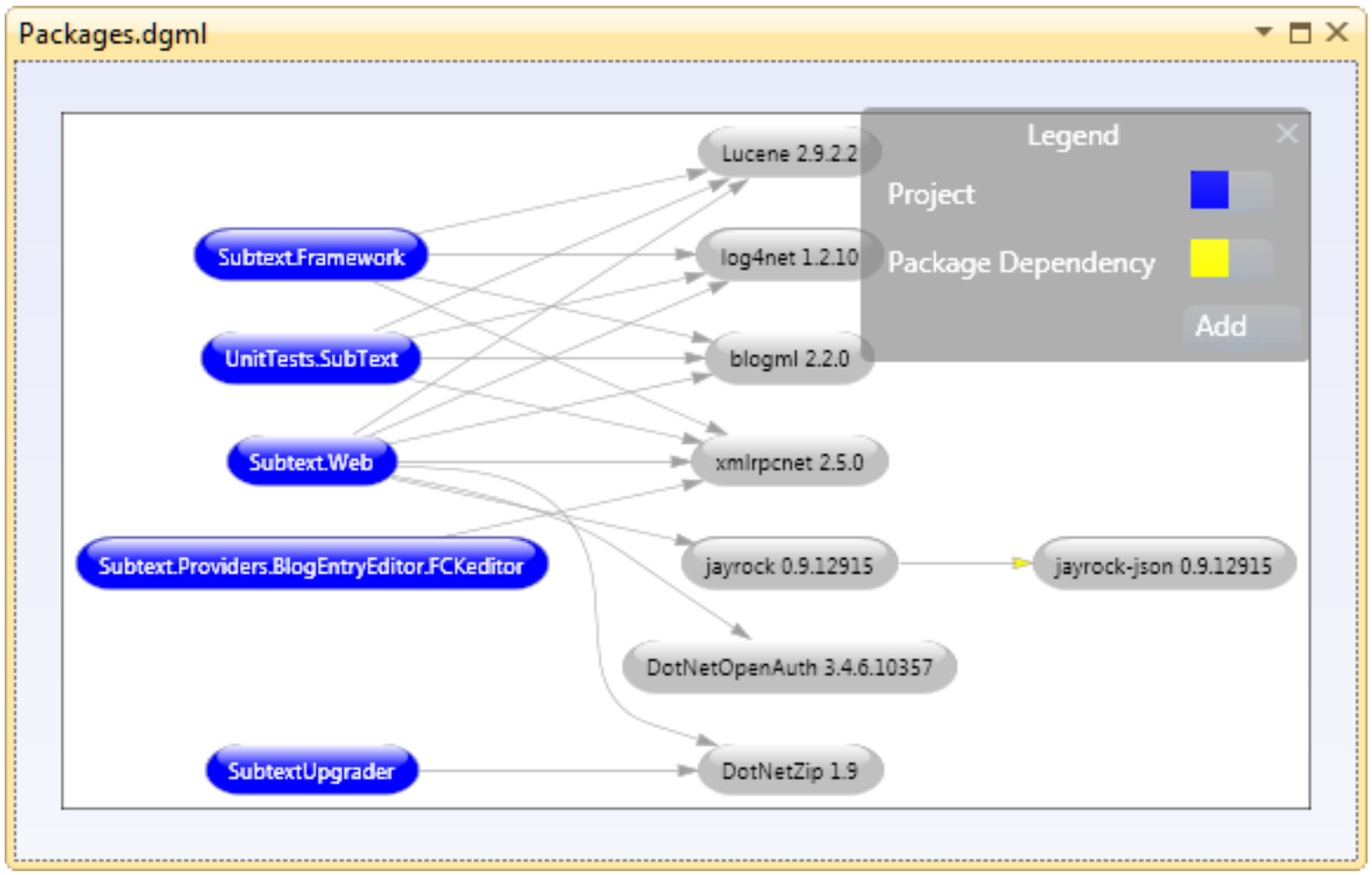
Owen Mundy | Fall 2013

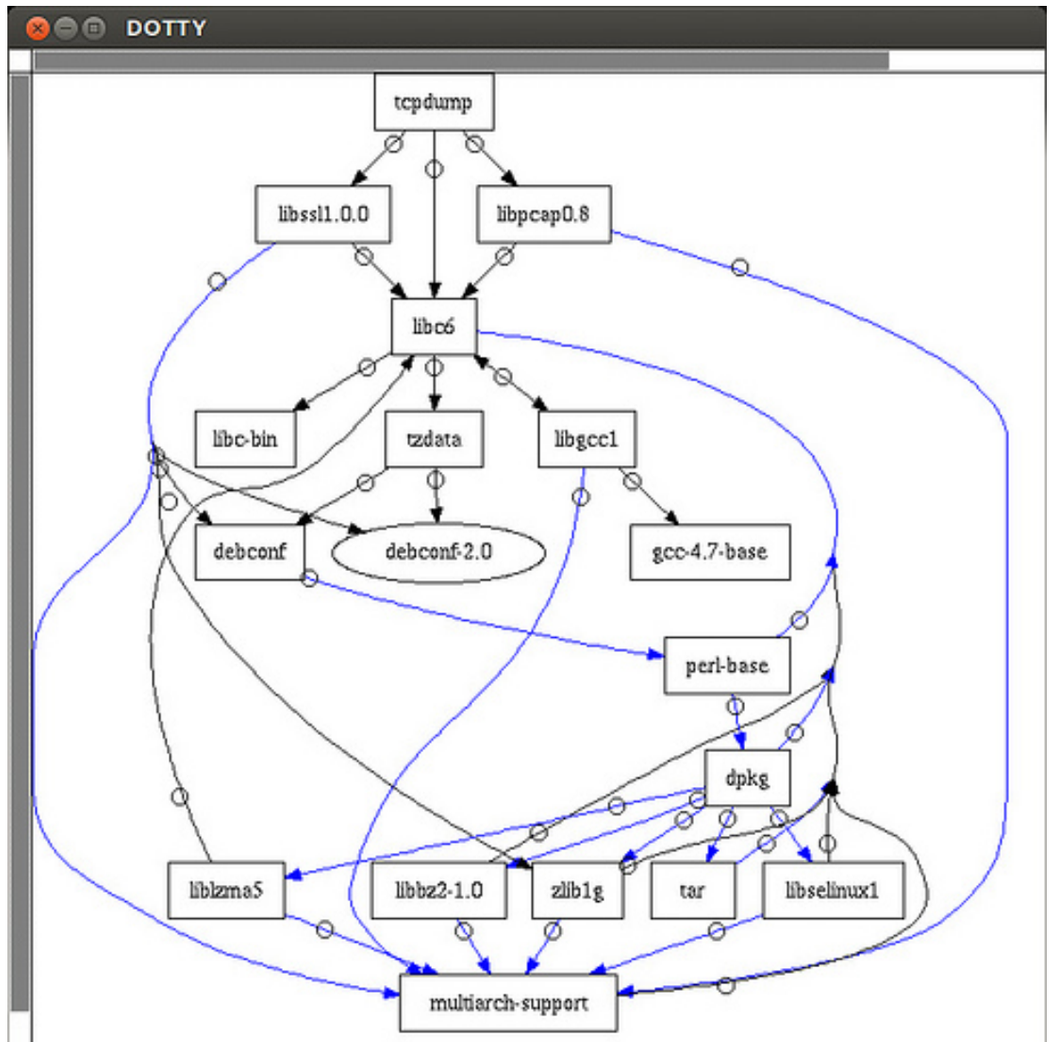
Overview

- Introduction to Linux package managers
- Installing libraries
- Introduction to GPIO pins
- How to find your revision number
- How to use crimpers
- Accessing GPIO pins with Python
- More Python
- Cron jobs

Package Managers

- A package manager is a collection of software tools to automate the process of installing, upgrading, configuring, and removing software packages.
- They maintain a database of software dependencies and version information to prevent software mismatches and missing prerequisites.
- They are designed to save time by eliminating the need for manual installs and updates, as well as managing shared dependencies.





Package Managers

- Raspbian (and Debian) uses APT (Advanced Packaging Tool) which is actually a collection of tools. This includes **apt-get** which retrieves and installs software and dependencies. Let's use it now to **update** our system:

```
$ sudo apt-get update
      |         |         |
superuser   tool    command
```

owenmundy — pi@robomonitor2: ~ — ssh — ttys000 — 80×24

```
Hit http://raspberrypi.collabora.com wheezy Release.gpg
Hit http://mirrordirector.raspbian.org wheezy Release.gpg
Hit http://archive.raspberrypi.org wheezy Release
Hit http://raspberrypi.collabora.com wheezy Release
Hit http://mirrordirector.raspbian.org wheezy Release
Hit http://archive.raspberrypi.org wheezy/main armhf Packages
Hit http://raspberrypi.collabora.com wheezy/rpi armhf Packages
Hit http://mirrordirector.raspbian.org wheezy/main armhf Packages
Hit http://mirrordirector.raspbian.org wheezy/contrib armhf Packages
Hit http://mirrordirector.raspbian.org wheezy/non-free armhf Packages
Hit http://mirrordirector.raspbian.org wheezy/rpi armhf Packages
Ign http://archive.raspberrypi.org wheezy/main Translation-en_GB
Ign http://raspberrypi.collabora.com wheezy/rpi Translation-en_GB
Ign http://archive.raspberrypi.org wheezy/main Translation-en
Ign http://raspberrypi.collabora.com wheezy/rpi Translation-en
Ign http://mirrordirector.raspbian.org wheezy/contrib Translation-en_GB
Ign http://mirrordirector.raspbian.org wheezy/contrib Translation-en
Ign http://mirrordirector.raspbian.org wheezy/main Translation-en_GB
Ign http://mirrordirector.raspbian.org wheezy/main Translation-en
Ign http://mirrordirector.raspbian.org wheezy/non-free Translation-en_GB
Ign http://mirrordirector.raspbian.org wheezy/non-free Translation-en
Ign http://mirrordirector.raspbian.org wheezy/rpi Translation-en_GB
Ign http://mirrordirector.raspbian.org wheezy/rpi Translation-en
Reading package lists... 9%
```

Package Managers

- And now **upgrade** our system:

```
$ sudo apt-get upgrade
  |       |       |
superuser tool  command
```

- Note all the dependencies below (that we didn't have to install :-)

owenmundy — pi@robomonitor2: ~ — ssh — ttys000 — 80×24

```
Setting up libwbclient0:armhf (2:3.6.6-6+deb7u1) ...
Setting up libsmbclient:armhf (2:3.6.6-6+deb7u1) ...
Setting up samba-common (2:3.6.6-6+deb7u1) ...
Setting up smbclient (2:3.6.6-6+deb7u1) ...
Setting up libsystemd-login0:armhf (44-11+deb7u4) ...
Setting up apt-utils (0.9.7.9+rpi1) ...
Setting up python-minimal (2.7.3-4+deb7u1) ...
Setting up python (2.7.3-4+deb7u1) ...
Setting up idle (2.7.3-4+deb7u1) ...
Setting up libdpkg-perl (1.16.12) ...
Setting up dpkg-dev (1.16.12) ...
Setting up libgs9-common (9.05~dfsg-6.3+deb7u1) ...
Setting up libgs9 (9.05~dfsg-6.3+deb7u1) ...
Setting up ghostscript (9.05~dfsg-6.3+deb7u1) ...
Setting up linux-libc-dev:armhf (3.2.51-1+rpi1) ...
Setting up python-crypto (2.6-4+deb7u3) ...
Setting up dmsetup (2:1.02.74-8+rpi1) ...
update-initramfs: deferring update (trigger activated)
Setting up libdevmapper1.02.1:armhf (2:1.02.74-8+rpi1) ...
Setting up libdevmapper-event1.02.1:armhf (2:1.02.74-8+rpi1) ...
Setting up liblvm2app2.2:armhf (2.02.95-8+rpi1) ...
Processing triggers for menu ...
Processing triggers for initramfs-tools ...
pi@robomonitor2 ~ $
```

Installing Python libraries

- We can also APT to install other tools which help manage the system. Now we're going to install some libraries and a special package manager for Python.
- Install **setuptools** (download, build, install, upgrade, uninstall Python packages)

```
$ sudo apt-get install python-setuptools
```

- Install **pip** (a tool for installing and managing Python packages from the Python Package Index)

```
$ sudo easy_install pip
```

- Add the latest dev packages for Python (2.x)

```
$ sudo apt-get install python-dev
```

- And finally we'll install the **Raspberry Pi GPIO library**

```
$ sudo pip install rpi.gpio
```

Confirm RPi.GPIO installed

- Let's make sure the Python GPIO library installed correctly. Run the Python interpreter as a superuser so we will have access to pins:

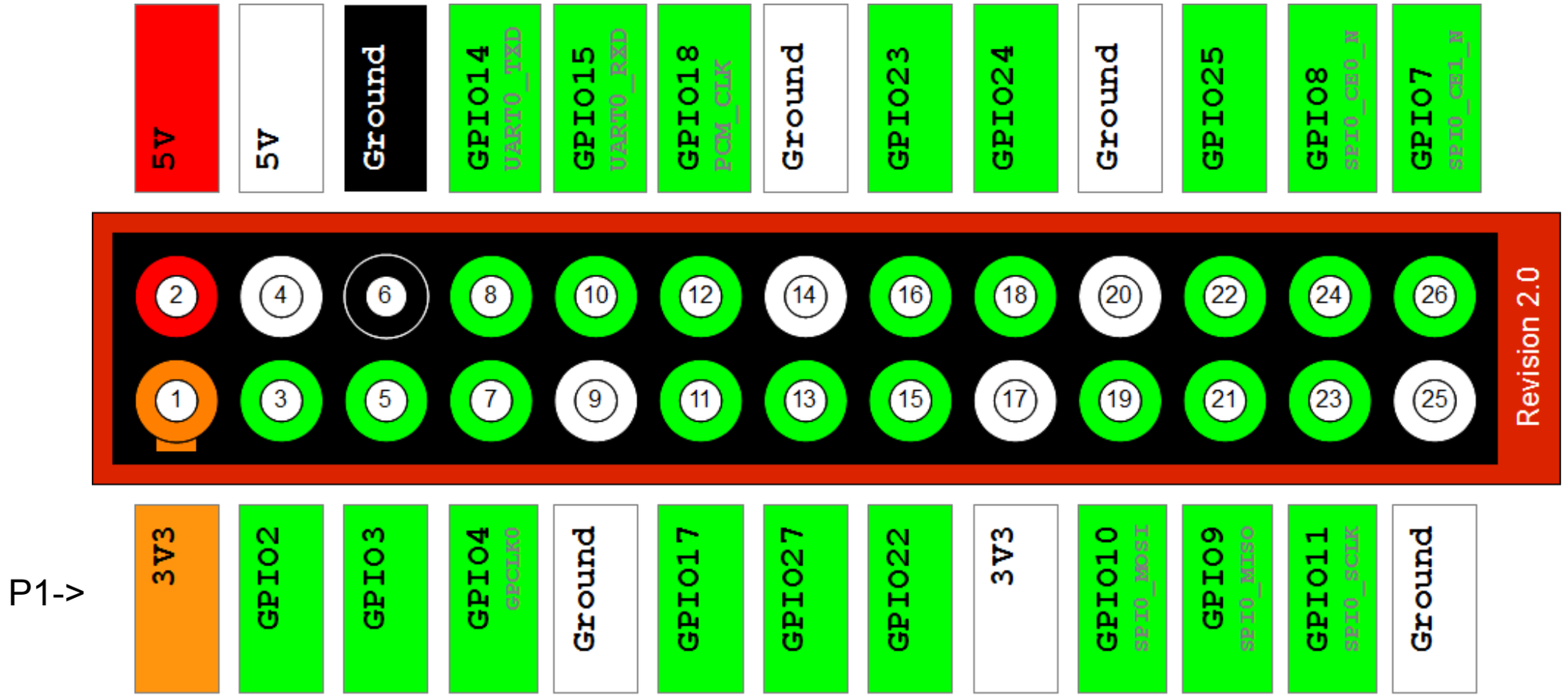
  $ sudo python

  # Now at the Python prompt (>>>) type:
  import RPi.GPIO as GPIO

- You should just see the prompt again (>>>) which means the command executed without errors and the library imported correctly.

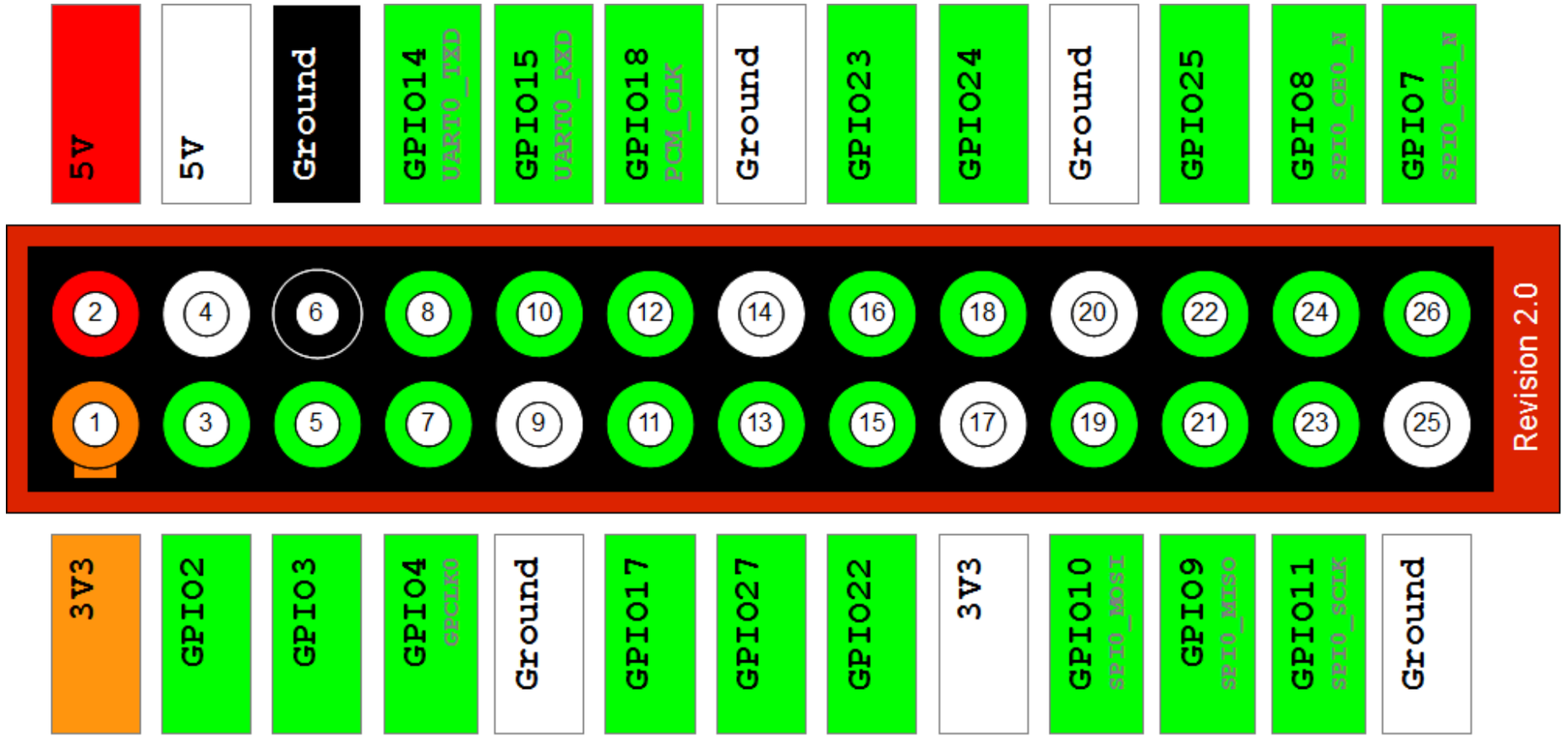
GPIO (General Purpose Input/Output) pins

- The 26 GPIO pins on the Raspberry Pi are available on the PCB via a header and allow you to interface the Pi to the real world.
- It is only safe to use the 3.3V. You should never measure 5V with a GPIO pin.
- You should always unplug your Pi before making any changes to your wiring.



Raspberry Pi GPIO numbering

- The two ways of IO pin numbering **BOARD** numbering and **BCM**.
- **BOARD** numbering refers to the pin *numbers* on the P1 header. The advantage is that your hardware will always work. Regardless of your board revision, you will not need to rewire your connector or change your code.
- **BCM** numbers are a lower level way of working. It refers to the channel numbers on the Broadcom SOC, so you must reference a diagram when you plug into the board, and your script could break between revisions of Raspberry Pi boards.



Source: RPi.GPIO module basics

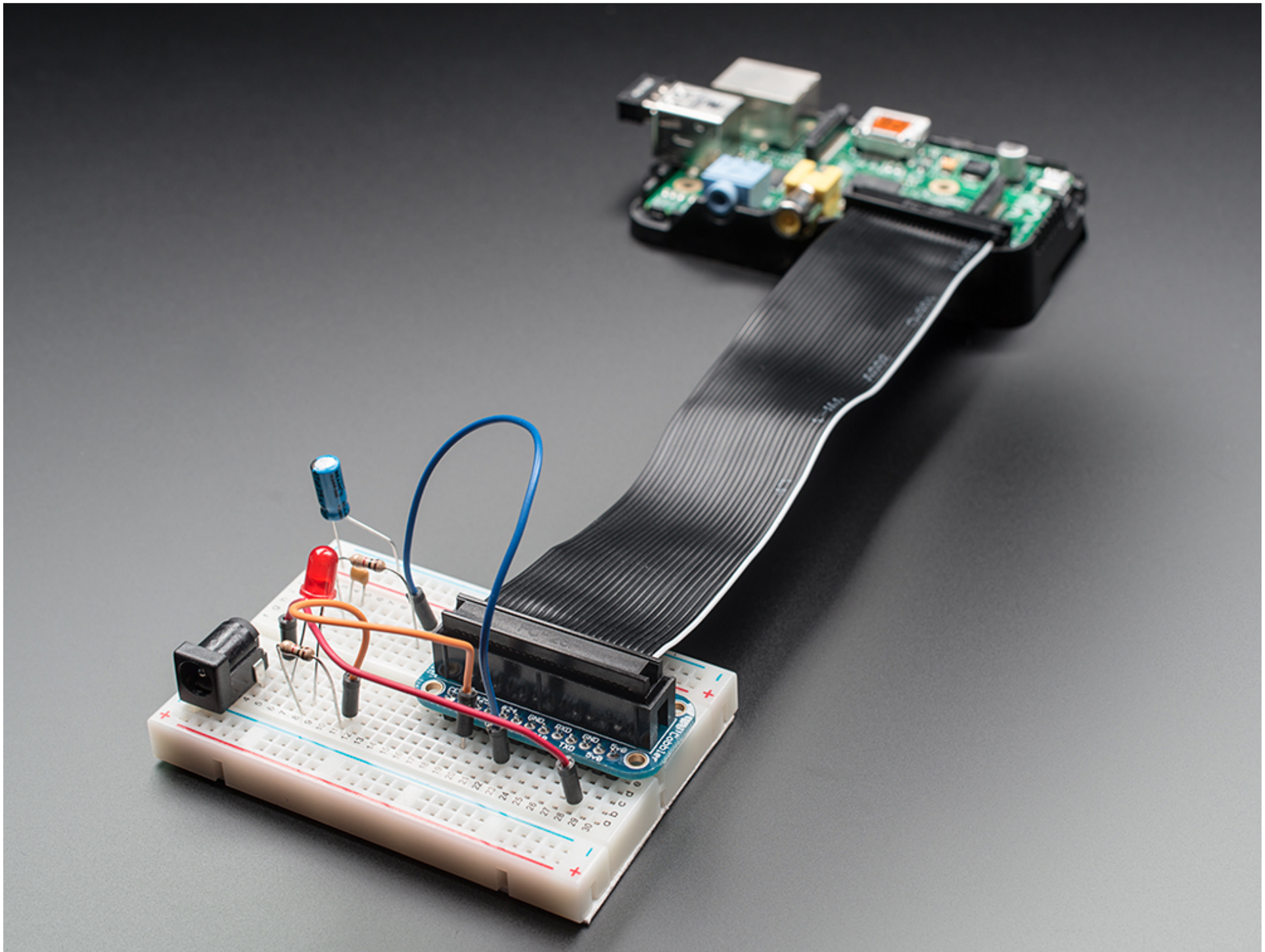
How to find your revision number

- A few different Pis have been released already. Your Pi could be model A or B, and it could be one of a number of revisions w/in that model which includes changes to mounting holes, the power supply circuitry, and GPIO pins.
- To find out your revision number type:
  $ cat /proc/cpuinfo
- And use this chart:

Model and Pi Revision	/	Hardware Revision Code from cpuinfo
Model B Revision 1.0	/	0002
Model B Revision 1.0 + ECN0001 (no fuses, D14 removed)	/	0003
Model B Revision 2.0	/	0004, 0005, 0006

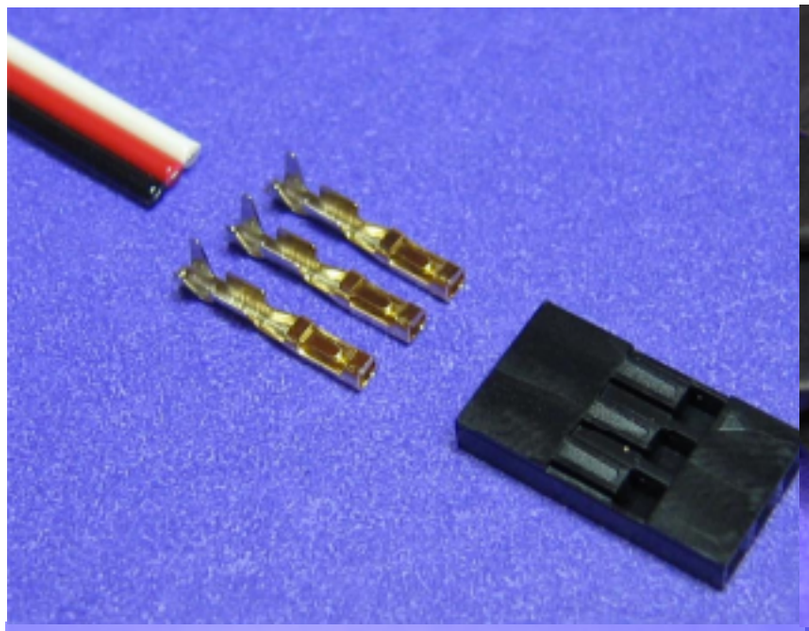
Connecting headers

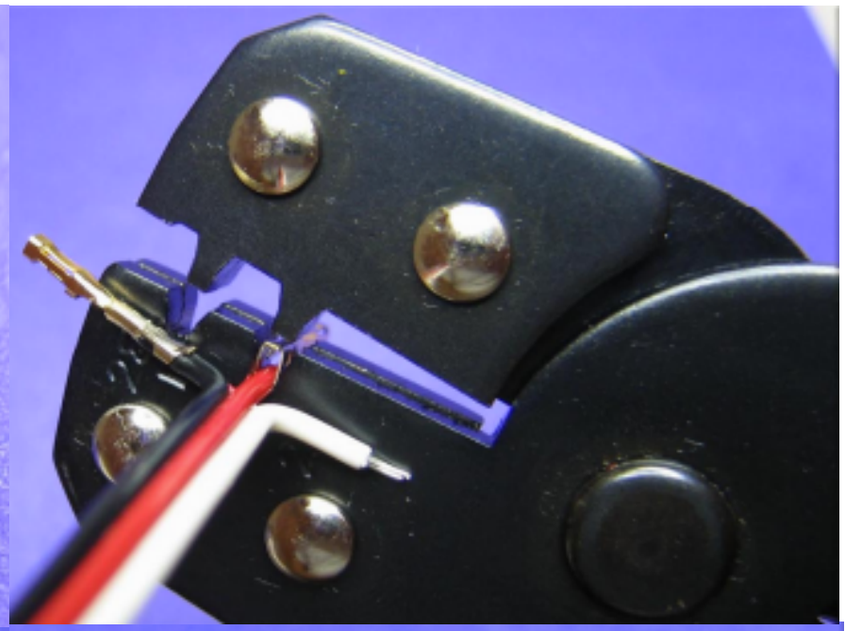
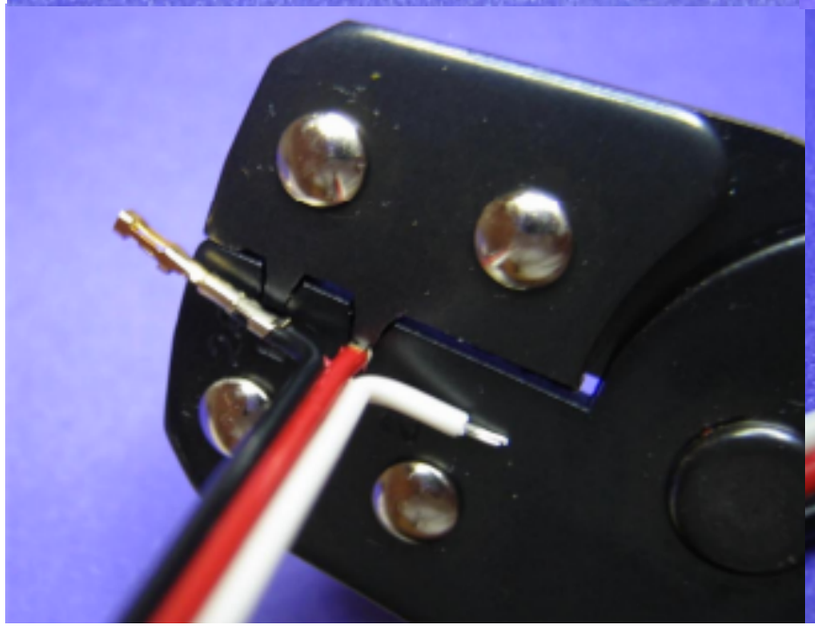
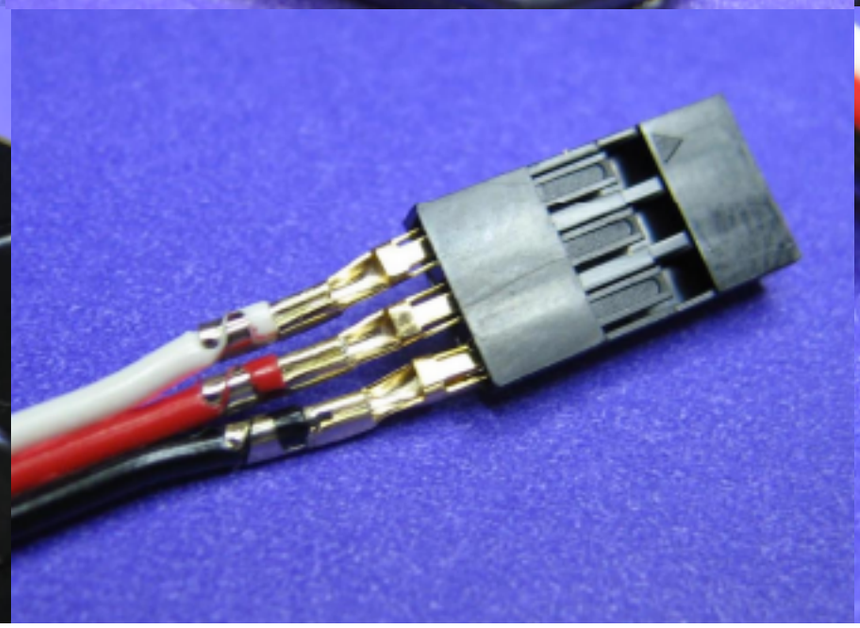
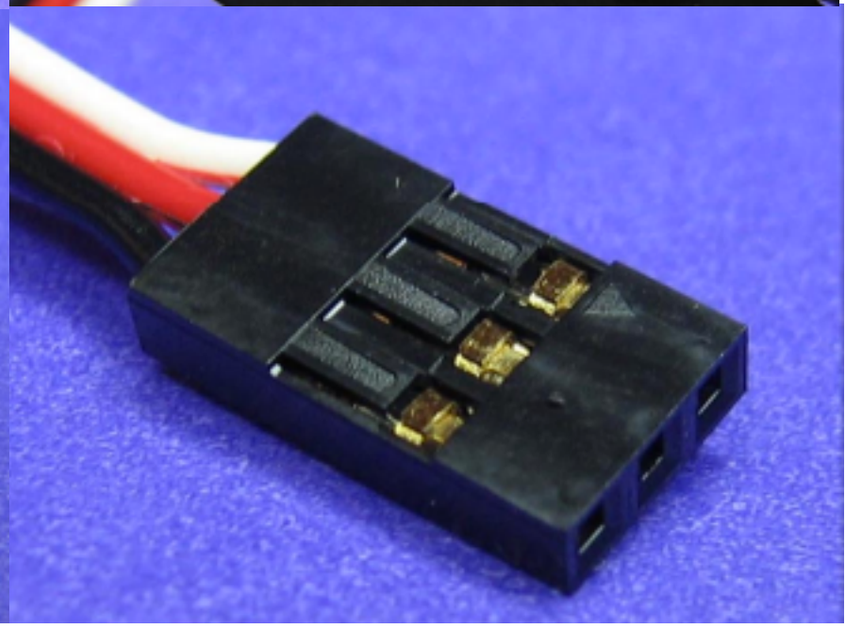
- To use GPIO pins we need to connect them to a breadboard somehow with wire and female headers. We could use Lady Ada's $8 Pie Cobbler Breakout+Cable.



How to use crimpers

- Or we could make our own!

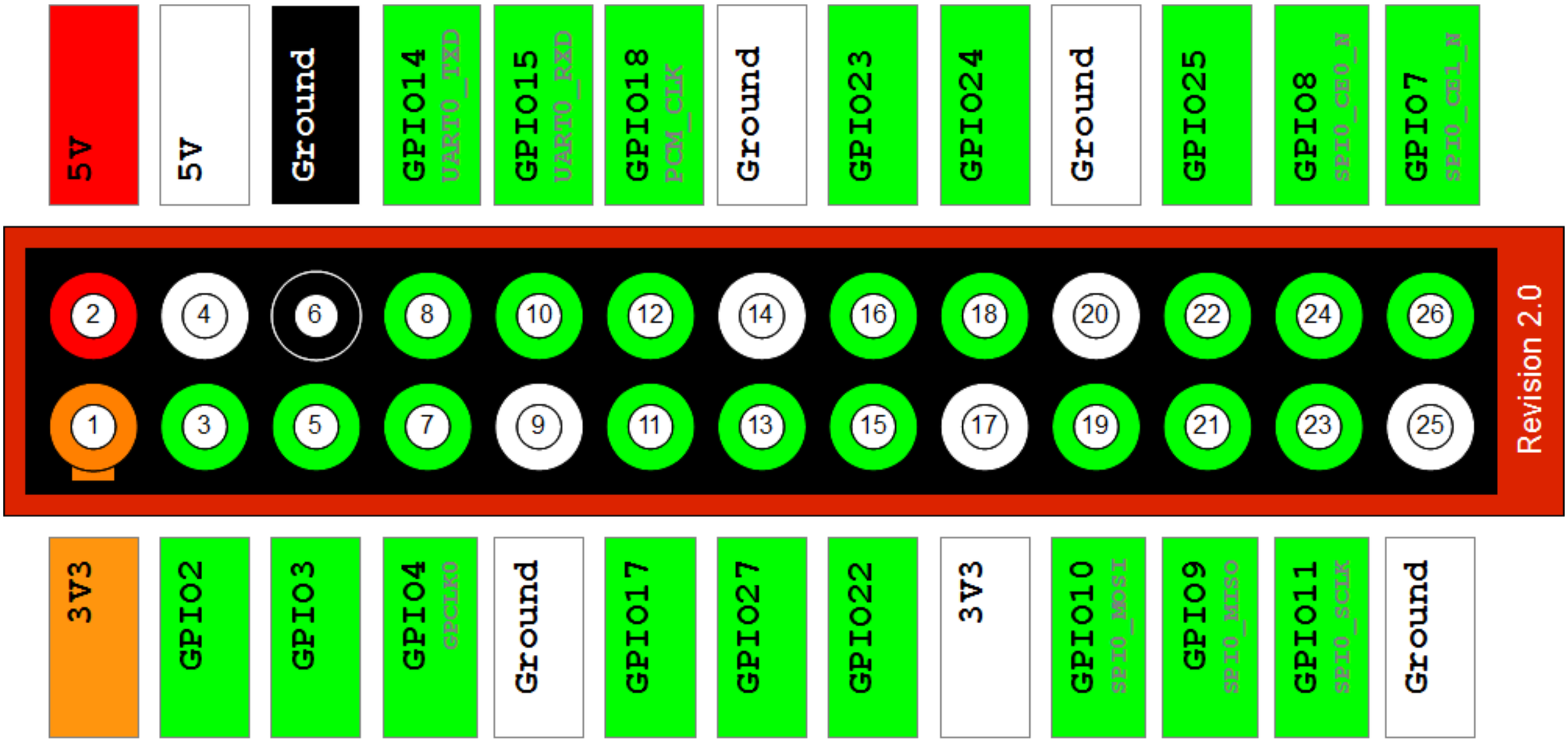


- See: Instructions (PDF) or Crimping a Servo Connector (3:38)

Raspberry Pi GPIO

- Connect an LED and resistor in series to pin 8 (GPIO14) then the other end to ground using the connector we build in class.
- Here's the layout for reference.



Hello World! with Raspi GPIO and Python

- Make a new file inside ~/ called blink.py ...

```
# import the library
import RPi.GPIO as GPIO

# tell Python to use Raspberry Pi board pins
GPIO.setmode(GPIO.BOARD)

# setup channel 8 as an output
GPIO.setup(8, GPIO.OUT)

# turn the LED on and off
GPIO.output(8,True)
GPIO.output(8,False)

# time for cleanup
GPIO.cleanup()
```

Hello World! with Raspi GPIO and Python

- Save it and then in the command line cd into the directory where the file is:

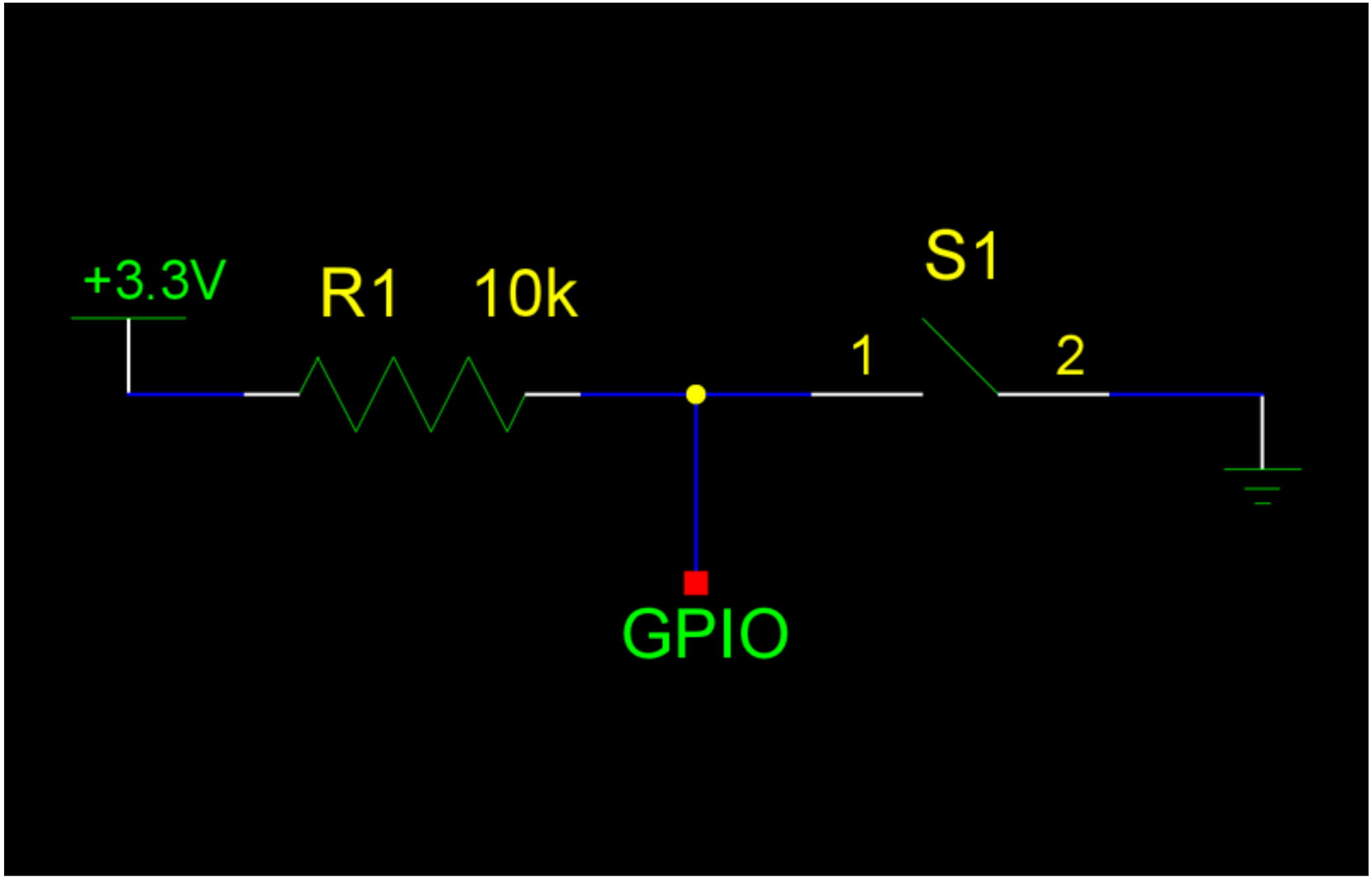
  $ cd ~/

- Then use python to execute the file:

  $ sudo python blink.py

Using a button with Raspi

- Now let's use GPIO.input to read the value of a button.
- Wire the switch like you see here...



Source: adafruit learning system

Using a button with Raspi

- Make a new file inside ~/ called button.py ...

```
# import library
import RPi.GPIO as GPIO

# set header pin numbers
GPIO.setmode(GPIO.BOARD)

# set pin 8 as input
GPIO.setup(8,GPIO.IN)

# start a continuous loop
while True:
    if (GPIO.input(8) == False):
        print("Button Pressed")

# cleanup
GPIO.cleanup()
```

Source: RPi.GPIO module basics

Using a button with Raspi

- When you use python to execute the file you will find it prints over and over while the button is down.

  $ python button.py

- We can fix this easily by adding a pause in the script. Import time at the top:

  import time

- And then inside the if statement use the sleep() method to pause for one second:

  time.sleep(1)

Source: RPi.GPIO module basics

Create a cron job on the Raspberry Pi

- Make a new file inside ~/ called blink.py ...

```
# import libraries:
import RPi.GPIO as GPIO
import time

# use board pin numbering and set pin 8 to OUT
GPIO.setmode(GPIO.BOARD)
GPIO.setup(8, GPIO.OUT)

# define a function named Blink()
def Blink():
    GPIO.output(8,True) # on
    time.sleep(2) # wait
    GPIO.output(8,False) # off
    time.sleep(2) # wait
    print "Done"
    GPIO.cleanup()

Blink()
```

Create a cron job on the Raspberry Pi

- Edit your crontab file (create one if it doesn't exist)

  $ crontab -e

- Type (see Configure time in crontab) to make a script run every minute:

  * * * * * sudo python /home/pi/blink.py

- Or every ten minutes:

  */10 * * * * sudo python /home/pi/blink.py

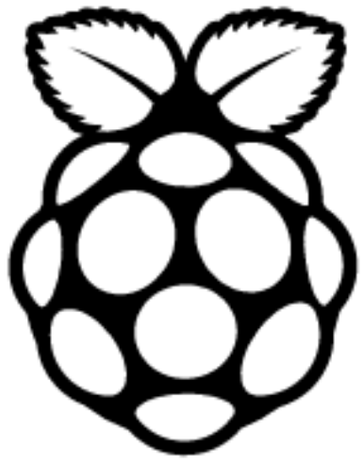
- Or, start a script on boot (like for button detection):

  @reboot sudo python /home/pi/button.py &

  ^ The "&" at the end of the line means the command is run in the background and it won't stop the system booting up.

Raspberry Pi 3

Recipes



Owen Mundy | Fall 2013

Overview

- Read a photo resistor with Raspberry Pi
- Read a temperature sensor with Raspberry Pi
- Capture images from a USB webcam and upload them to a server
- Capture images / video from a PiCam with raspistill / raspivid
- Use a thermal printer with the Raspberry Pi

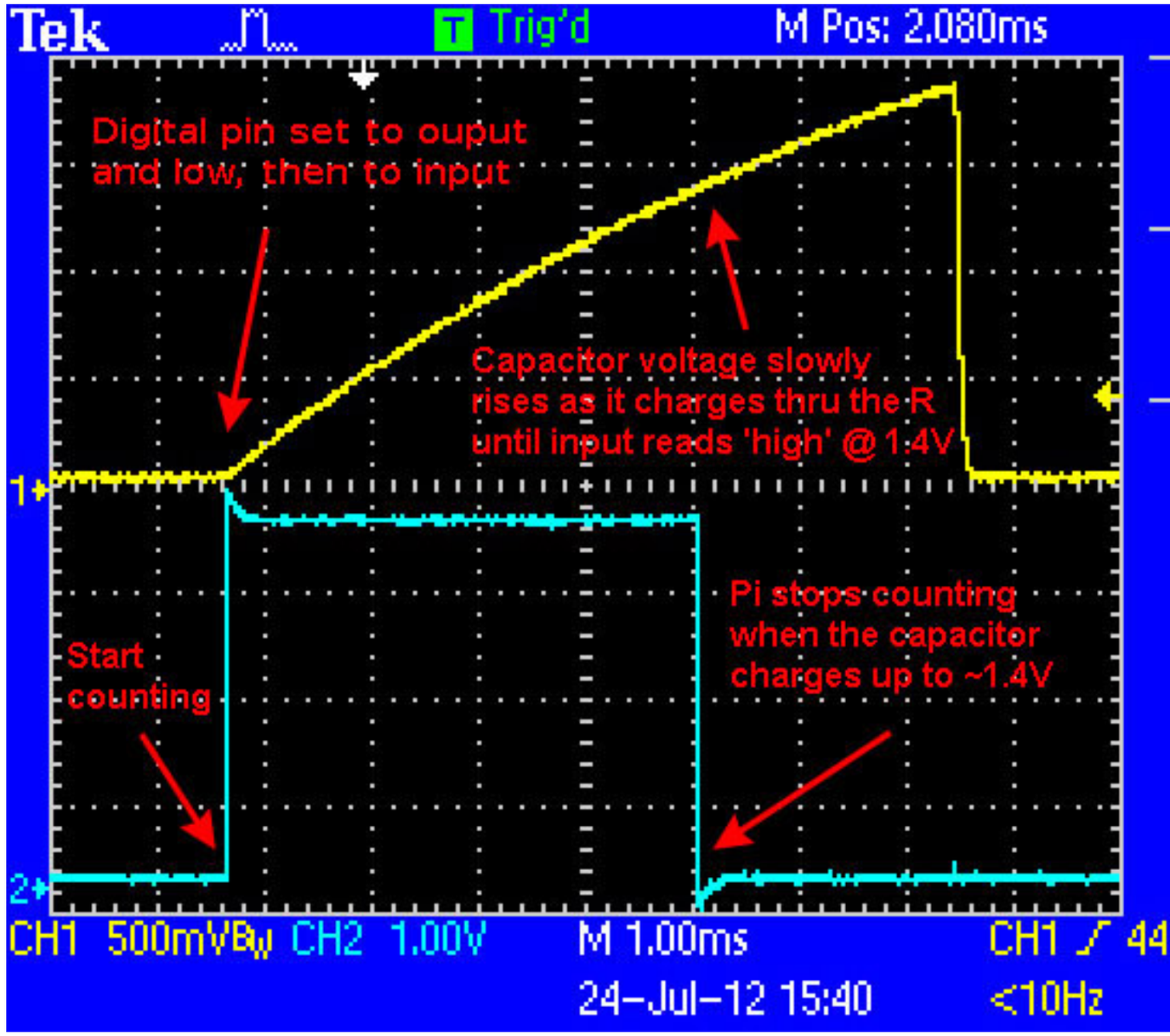
Read a photo resistor with Raspberry Pi

- The most precise way to read analog sensors with the Pi is with an Analog-to-Digital Converter (ADC) chip.
- A less precise method is to keep track of the amount of time it takes to fill up or “charge” a capacitor.
- The drawback is that it depends on the Pi timing itself, which can vary based on how 'busy' the computer is.
- This technique only works with sensors that act like resistors like: photocells, thermistors (temperature sensors), flex sensors, force-sensitive resistors, and many more.
- It cannot be used with sensors that have a pure analog output like IR distance sensors or analog accelerometers. You will need a real ADC for them.

Source: adafruit learning system

Read a photo resistor with Raspberry Pi

- This capture from an oscilloscope shows the digital pin (yellow) and the Pi starts and stops counting (blue) about 4.5ms later.
- The capacitor is like a bucket and the resistor is like a thin pipe.
- In this case, our 'bucket' is a 1uF ceramic capacitor. You can change the capacitor nearly any way you want but the timing values will also change. 1uF seems to be an OK place to start for most sensors.
- If you want more range, use a bigger cap - but it will take longer to measure. For faster reads, go with a smaller capacitor

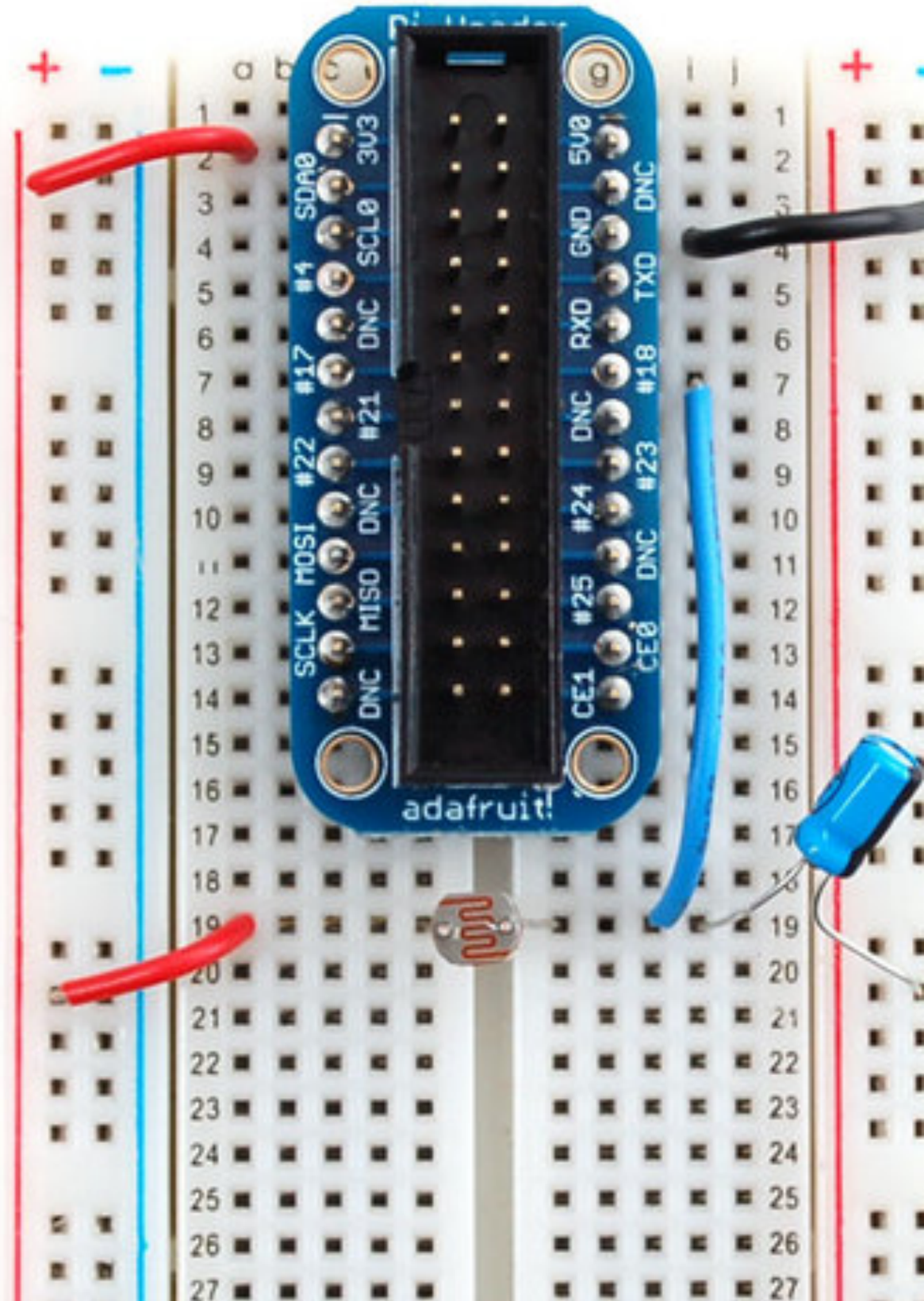


Source: adafruit learning system

Read a photo resistor with Raspberry Pi

- Hook up your Pi like this:

GPIO #18
|
3.3V --- LDR --- 1uF capacitor --- GND



Source: adafruit learning system

Read a photo resistor with Raspberry Pi

- Inside lightsensor.py ...

```
import RPi.GPIO as GPIO, time, os
DEBUG = 1
GPIO.setmode(GPIO.BCM)

def RCtime (RCpin):
        reading = 0
        GPIO.setup(RCpin, GPIO.OUT)
        GPIO.output(RCpin, GPIO.LOW)
        time.sleep(0.1)
        GPIO.setup(RCpin, GPIO.IN)
        # This takes about 1 millisecond per loop cycle
        while (GPIO.input(RCpin) == GPIO.LOW):
                reading += 1
        return reading

while True:
        print RCtime(18)     # Read RC timing using pin #18
```

Source: adafruit learning system

Capture images from a USB webcam

- The Raspberry Pi has two USB ports which can be used for all kinds of things, including cameras.
- In this recipe we're going to install fswebcam and take a picture.

```
# install
sudo apt-get install fswebcam

# view help
fswebcam --help
```

Capture images from a USB webcam

- You can see what cameras work well with Linux here:
- Once you have plugged your camera in...

```
# list cameras
ls /dev/video*

# now take a photo referencing the appropriate port
fswebcam -r 960x720 -d /dev/video0 image.jpg

# fswebcam can write on photos!
fswebcam -r 960x720 -d /dev/video2 --jpeg 80 --title "hello world!" --line-colour
"#FF000000" --banner-colour "#AA000000" image.jpg
```

- Also see: fswebcam tutorials
  Raspberry Pi Webcam; a Gentle Intro to Crontab
  Ubuntu fswebcam manual
  fswebcam on Github

More Raspberry Pi tutorials

- http://www.raspberrypi-spy.co.uk/
- http://www.cl.cam.ac.uk/projects/raspberrypi/tutorials/robot/image_processing/
- http://geekgurldiaries.blogspot.com/search/label/Raspberry%20Pi
- http://learn.adafruit.com/category/raspberry-pi