Raspberry Pi/Raspian/Python Notes

Last edited: 22-Jul-17

Contents

[Raspberry Pi/Raspian/Python Notes 1](#_Toc485925649)

[Packages I installed (name, source, special instructions) 10](#_Toc485925650)

[Physical interface APIs 11](#_Toc485925651)

[Python Notes 12](#_Toc485925652)

[Git Notes 15](#_Toc485925653)

[GrovePi+ Notes 15](#_Toc485925654)

This document is a valuable tool for me to learn and reference specific findings I have made in the Raspberry Pi domain. As this domain is undergoing constant change and because I have a lot to learn, this document is a work in process. I also need to have the document available on my primary PC, a laptop, and even a Raspberry Pi computer, I plan to keep a master under version control on GitHub in a 'Notes' project. Document updates will be updated on the current computer in use and committed to the GitHub master repository.

These notes originated in 2015 for use on a Raspberry Pi 2 Model B v1.1 using the latest Raspian distribution package at the time. The current notes reflect changes that have evolved since then. If notes don't seem to fit what is viewed on the RPi screen, suspect a revision in the ever evolving software in use.

Topics:

1. Installing operating system images. Start here:

<https://www.raspberrypi.org/documentation/installation/installing-images/README.md>

and here:

<https://www.raspberrypi.org/documentation/installation/installing-images/windows.md>

and furthermore: NOOBS Setup at

<https://www.raspberrypi.org/help/noobs-setup/>

Note Win32DiskImager is already installed. Look for application named "Image Writer".

**Configuration list (for new systems on SD memory.)**

* 1. Using NOOBS on newly formatted SD memory card
     1. Specify WiFi router pass code.
     2. Install Raspbian.
  2. In GUI Preferences, use RPi Configuration tool to enable camera, Set Localization items.
  3. Copy key development files from old SD memory to new SD card.
  4. Perform sudo apt-get update & sudo apt-get upgrade, installs for camera utilities, GPIO. (Examples: beware some instructions may be obsolete or buggy as experienced in raspberrypi.org)

Camera: <https://www.raspberrypi.org/documentation/configuration/camera.md>

1. RPi control through the command line interface (CLI).
   1. Connecting a keyboard/monitor to the RPi and booting into the CLI environment. Also, later using a remote laptop as terminal running a terminal emulator (PuTTY).
      1. Log-in using username/password. Default is "pi"/"raspberry"
      2. Type BASH commands to perform desired operations in CLI.
   2. Starting the Raspian GUI.
      1. Enter *startx* on the command line followed by Enter. Automatic entry into the GUI may occur if Preferences.RaspberryPiConfiguration is configured to do this.
   3. Shutting down safely.
      1. From the GUI.
         1. Click on File.Exit.Shutdown.
      2. From the CLI. *sudo shutdown -h now*
         1. For hardware/Python method visit:
   4. Mounting a USB thumb drive to transfer files.
      1. CLI approach
      * Find out what the USB drive is called: sudo fdisk -l (For this example returns */dev/sda1*)
      * Create a mount point: sudo mkdir /media/usb
      * Mount: sudo mount */dev/sda1* /media/usb
      * Do work (copy files two or from USB drive)
      * Unmount: sudo umount /media/usb
      * Credit <http://askubuntu.com/questions/37767/how-to-access-a-usb-flash-drive-from-the-terminal-how-can-i-mount-a-flash-driv>.
      * Error saying disk is already mounted? Use df –h to get list of mounted drives and paths. Study the paths to find where to access the thumb drive.
      1. GUI approach
      * Plug USB drive in open slot.
      * Use File Explorer to find USB contents in /media/pi/root
      * Do stuff (copy from-to)
      * Eject from icon in upper right of GUI. (Click on triangle-over-bar icon to get a list of media along with their folders in /media/pi (boot, root, SETTINGS)

1. Configuration of Raspbian:
   1. From the CLI - sudo raspi-config
   2. From the GUI - See Preferences
2. Display a list of running processes: *ps ax*
3. Display the current SSH version: *ssh -V*
4. Other key command line commands: See [Debian Refcard.pdf](DebianRaspian%20OS/Debian%20Refcard.pdf)
5. Capturing history of all command lines to file: history 1>historyfile.txt See [redirection](http://www.tldp.org/LDP/abs/html/io-redirection.html) for more details.
6. Taking a screen shot in CLI: *scrot*.
   1. scrot with x second delay: scrot *-d x*
   2. scrot with countdown delay: scrot *-cd x*
   3. scrot for currently active window: scrot *-u -cd x*
   4. *scrot to output file (date, time, resolution, scrot) and move to receiving file:*
   5. *scrot '%Y-%m-%d*\_%H%M *\_$wx$h\_scrot.png' -e 'mv $f ~/images/shots/'*
7. Locating packages for my unique needs:
   1. Look in Debian Packages for jessie: <https://packages.debian.org/stable/>
   2. To search available packages for a keyword, use 'search' plus the keyword of interest:

*apt-cache search <keyword>* or *apt-cache pkgnames <specific package name>*

for long lists pipe to grep*: | grep <keyword> |more*

1. Finding and Installing packages: *sudo apt-get install <package name>*. Optionally, add '*-Y*' to automatically say Yes to questions that may arise.
2. List installed packages:
   1. *dpkg --get\_selections > dpkg-selections.txt* (OR *| less*),
   2. OR *dpkg --list* (mentioned elsewhere in this document),
   3. OR *aptitude* (user friendly cli)
   4. Alternate: [synaptic](https://thepihut.com/blogs/raspberry-pi-tutorials/16998732-finding-out-what-is-installed-on-your-pi-and-installing-more) (install with *sudo apt-get install synaptic*)
3. Removing packages:
   1. *sudo apt-get purge <package name>* to remove main package
   2. *sudo apt-get autoremove* to uninstall dependencies that were added with the purged package.
4. Creating and running Shell Scripts. See [Advanced Bash-Scripting Guide](http://www.tldp.org/LDP/abs/html/why-shell.html). Also, see snappic.sh example below.
   1. First line is required, this is called the 'shebang':

#! /bin/bash

* 1. Make script executable: *chmod +x <script name>.sh*

1. Running from the command line should be done by typing "./snappic.sh" if already in the folder holding the script. Leaving the ./ prefix off doesn't run the script! When script development is complete, copy it with a "sudo cp" to /usr/local/bin to eliminate "./" requirement. See [Invoking the Script](http://www.tldp.org/LDP/abs/html/invoking.html).
2. Scripts can invoke other scripts as long as the absolute path is defined (or it is in /usr/local/bin)
3. The script can run on a schedule using crontab instructions below. No further interaction needed from power-up.
4. Saving data in the cloud: See examples for [Dropbox Sync](https://learn.adafruit.com/cloud-cam-connected-raspberry-pi-security-camera/dropbox-sync) and [Adafruit IO](https://learn.adafruit.com/cloud-cam-connected-raspberry-pi-security-camera/adafruit-io) on adafruit website. *Dropbox Uploader notes*: Have used shell file with dropbox\_uploader call to save images on DB as scheduled by cron (See MyProjects/snappic.sh for the bash script). If there is a way to embed in Python code, I have yet to find it. When installing Dropbox Uploader, it will be put your Documents directory in its own folder as: /home/pi/Documents/Dropbox-Uploader. For additional guidelines, see other Dropbox setup instructions in the Dropbox Sync link above. Following these instructions, you can create a unique Dropbox folder for each RPi instance or give yourself access to all your DB folders. For organization purposes, the following installations are recorded:

Media App name in Dropbox Dropbox Folder

* 1. ~~MicroSD 32GB for RPi 3 RPi3Uploader (?) Dropbox/Apps/RPiPictures~~ (erased when GrovePi OS was installed?!)
  2. MicroSD 32GB for RPi2B RPi2B8977 Full Dropbox access. Default to Dropbox/
  3. MicroSD 32GB for RPi3B RPi3Uploader8977 Full Dropbox access, default to Dropbox/
  4. /MicroSD 8GB for RPi ZERO v1.3 RPiJunk Dropbox/Apps/RPiJunk

Type *dropbox\_uploader* on CLI for usage instructions. Sample upload from CLI when Full Dropbox access is used (this is a project folder backup): *dropbox\_uploader upload ~/Documents/PythonProjects/ /Apps/RPiPictures/* (trailing '/' is important). If uploading a file by its name, the filename must be provided for source and destination parameters. Example *download* of a file from Dropbox using the CLI. (the working directory is ~/Dropbox-Uploader): *./dropbox\_uploader.sh download ./Apps/RPiPictures/MyProjects/snappic.sh ~/Documents/PythonProjects/snappic.sh*

Another example of upload, from the project folder*: ~/Dropbox-Uploader/dropbox\_uploader.sh upload 'HTT\_Fri Dec 23 21:01:04 2016.csv' ./Apps/ If the file name has spaces, it must be enclosed in quotes (') or results are unpredictable. Or a*s a workaround, I used '\*' where the first space occurred in the filename and got several files uploaded in response. Also note the relative path to the *dropbox\_uploader.sh* script!

1. Installing the camera. This assumes RPi is booted into GUI vs CLI (Command Line Interface).

Background: <https://www.raspberrypi.org/documentation/usage/camera/> and

<https://www.raspberrypi.org/documentation/raspbian/applications/camera.md>

* 1. Start CLI window, a.k.a. "The Terminal".
  2. sudo apt-get update
  3. sudo apt-get upgrade
  4. exit to GUI
  5. Start Preferences.RaspberryPiConfiguration dialog from GUI and enable camera.
  6. Quit Config dialog. Reboot if necessary.

The camera can be used from Shell scripts or a Python program. For shell scripts, see: <https://www.raspberrypi.org/documentation/usage/camera/raspicam/README.md>

* 1. From Python 3, see examples in [RaspberryPi.org Getting Started with Pi Camera](https://www.raspberrypi.org/learning/getting-started-with-picamera/worksheet/). Coding is relatively easy! Also: <https://www.raspberrypi.org/documentation/usage/camera/python/README.md>
  2. From Shell script (created in CLI window): raspistill, raspivid, Time-Lapse, raspiyuv.
  3. For Time-Lapse collection see: <https://www.raspberrypi.org/documentation/usage/camera/raspicam/timelapse.md> for description of cron and crontab usage. Open for editing: crontab -e and the LeafPad editor for snappic.sh files.

Take a time-stamped picture with the shell script *snappic.sh* listed here*:*

#! /bin/bash

DATE=$(date +"%Y-%m-%d\_%H%M")

raspistill -o /home/pi/Pictures/$DATE.jpg

# Add feature to store images in Dropbox.

/home/pi/Dropbox-Uploader/dropbox\_uploader.sh upload /home/pi/Pictures/$DATE.jpg $DATE.jpg >&1

Use the LeafPad editor to open snappic.sh for editing. Folder Documents/MyProjects

Edit crontab to schedule snappic.h. Do crontab -e to open for edit and append this:

# m h dom mon dow command

# Camera Stills

\*/2 6-20 \* \* \* /home/pi/Documents/MyProjects/snappic.sh 2>&1

Which translates to:

"every 2 minutes, between 6am & 8pm (Standard time?), run snappic.sh, redirect stderr to stdout"

**To disable the time lapse script, comment out the crontab line containing snappic.sh with a "#".**

Set execute permission for scripts in MyProjects with the command:

chmod 777 -R /home/pi/Documents/MyProjects

or set Properties.Permissions for the folder in the GUI.

Can list crontab contents with the command: crontab -l

Get help with crontab with ^G after opening cronab -e.

1. Creating a time-lapse movie using RPi.

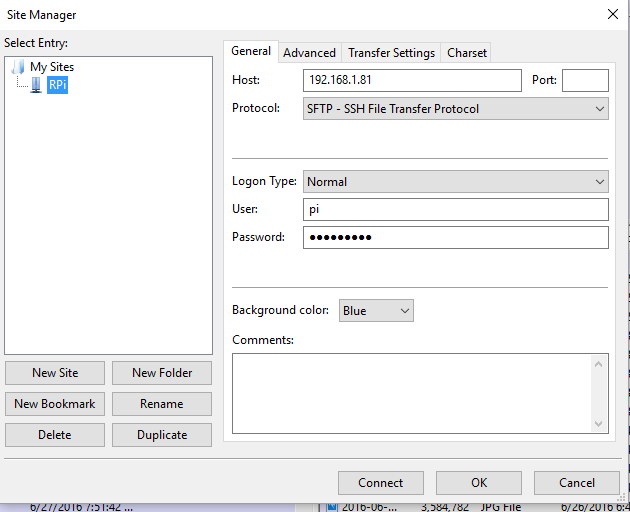
See Evernote>"TimeLapse Videos to iPad via a PC" for RPi notes.

1. Copy stills from microSD to C:\My Documents\Scratch.  Use FileZilla client on Windows PC or USB drive. (USB drive command is second best. Need to mount the drive on RPi with "sudo mount -t vfat /dev/sda1 /mnt" )

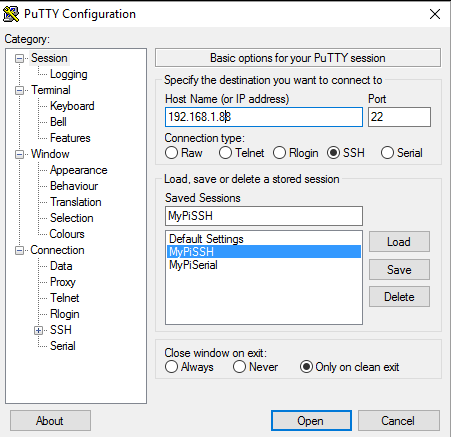
<http://trevorappleton.blogspot.com/2014/03/remotely-copy-files-to-and-from-your.html>

File.SiteManager screen looks like this: (note "Port" is blank) Host IP could be Ethernet or WiFi. If displayed, click through the warning of "unknown host key".

Host IP (for RPi) can be displayed in GUI window by hovering over the wireless icon in upper right corner. Note: the IP address may change if power has been recycled on the RPi!



1. Use Movie Maker (Windows Movie Maker, Version 2012 used here) to make .mp4 movie segments from batches of pictures (349 images in a batch has worked when 16GB RAM is available on the PC).  0.04 secs/frame.   Select and clear all stills after each batch.   Note where movie segments are stored: ThisPC/Videos. Reference: <http://www.instructables.com/id/How-To-Make-a-Film-with-Pictures-using-Windows-Mov/?ALLSTEPS>
2. If necessary, use Movie Maker to combine .mp4 movie segments.   Add segments to workspace, set same frame rate, & make movie.
3. Optionally, move output from This PC/Videos folder to Photos/ iPad Pics folder
4. Use iTunes to Sync Photos (including videos).
5. Watch video on iPad device.
6. Create an animated GIF using the ImageMagick package. This is described in Instructable *Raspberry Pi Class: Lesson 5 Navigate the RPI's Software: Part 2*
   1. Add these two lines to a shell script text file, <script name>.sh: 
   2. convert is a command from ImageMagick. Takes all saved jpegs withthe prefix of image and converts them into an animated GIF with a delay of 10/100 of a second and loops forever.
   3. Make the shell script executable with the *chmod +x <script name>.sh*
7. Wireless terminal connection to RPi (SSH). Multiple methods available depending on task and computer being used to connect to the RPi (PuTTY on PC, 'Juice' on Android, FileZilla on PC for file transfer.)
   1. This assumes the RPi target computer has ethernet or WiFi access.
   2. SSH (Secure Shell) is a function provided in the application PuTTY. This free .exe program is found on Pavilion at "C:\Users\Alan\Documents\Raspberry Pi\PuTTY\putty.exe".
   3. Obtain the target computer IP address. On my network this can be done with the FING App.
   4. Start PuTTY and enter the IP address where indicated. Select SSH access to the target. Click 'Open'. A typical configuration for PuTTY is illustrated here:



* 1. To end a SSH session in PuTTY: Just exit PuTTY using the X box inupper right. Other methods on web did not work for me: *Ctrl-D* and *~.* ***~.*** means 'disconnect' per man 1 ssh

1. Configuring a RPi (or ZERO) to run without keyboard or monitor ("headless"). See this excellent procedure [Raspberry Pi Zero Headless Setup](https://davidmaitland.me/2015/12/raspberry-pi-zero-headless-setup/), 1 Dec2015 on DavidMaitland.me. Used to configure Zero for wireless operation over SSH with Laptop. Procedure written for use on a Linux (RPi) system. To save space here, I converted the webpage to PDF and stored in Documents/Raspberry Pi/RPiHints/Raspberry Pi Zero Headless Setup.pdf.
2. **Configuring a RPi to start a python script on startup**. Run the program on boot with a special crontab command, *@reboot*. Alternatively, put a script in /etc/rc.local. This is said to work:

(sleep 10;python scriptname.py)&

The () makes enclosed commands run in the background. The '&' is also needed. Source: raspberrypi.org forum "Best way to run a python script at startup?"

**Alternate way to run a python script at startup**. If the python program writes to screen, this will not happen as currently documented. This is working on the GrovePi system ‘dex’ to start the High Tech Thermometer.

*/etc/rc.local* file has this line at the bottom: Note shebang is #! /bin/sh -e

#!/bin/sh -e

#

# rc.local

#

# This script is executed at the end of each multiuser runlevel.

# Make sure that the script will "exit 0" on success or any other

# value on error.

#

# In order to enable or disable this script just change the execution bits.

# Start data logging application, HTT

*sh /home/pi/Documents/PythonProjects/HTT/launcher.sh &*

exit 0

*launcher.sh* script file:

#!/bin/sh

#launcher.sh

#Navigate to home directory, then to this directory, then execute python script.

cd /

cd home/pi/Documents/PythonProjects/HTT

sudo python2 HTT\_Unified.py > /home/pi/Documents/PythonProjects/HTT/output.log 2> /home/pi/Documents/PythonProjects/HTT/error.log sudo python2

1. How to check if a program is running. 'ps' lists all running programs started from the command line. 'ps axg' lists all programs including all those owned by the system. **'ps axg | grep <prog name>'** filters out all in list that don't have the program name included.
2. What is loaded with this distribution of Debian > Raspian > Wheezy?
   1. dpkg --list
   2. dpkg --search <function>. <function> example: "python-rpi.gpio"
   3. *aptitude* is a friendlier package manager. Run 'aptitude' from command line to get to GUI. Review Help to get started.
3. **Transferring files between RPi and Windows**. Ans: [FileZilla](http://trevorappleton.blogspot.com/2014/03/remotely-copy-files-to-and-from-your.html). Use ifconfig to get RPi iP address. Fill Quickconnect fields for Host (RPi IP address), Username (pi), Password (raspberry) and Port (no entry) before clicking on Quickconnect button. Browse in the 'Remote site:' window to the folder holding files of interest, select one or more files, right-click and select
4. **Search disk for a file name**.Use the ‘tree’ command. For more info do ‘info tree’ or google up linux tree. Some useful commands.
   1. tree ~ : show all directories in my user account.
   2. tree ~ -d
   3. tree -P ‘snappic.sh’ –prune / : search entire disk for snappic.sh, don’t show folders without the file. NOTE prune has double hyphens which Word insists be shown as a long hyphen. “ is equivalent to ‘ in the options string. NOTE: there is a 'space' between "-P" and file string.
   4. Similarily: tree -P ‘cron\*’ / --prune NOTE: there is a 'space' between "-P" and file string. This displays all folders with files containing ‘cron’ and variations. No asterisk will look for exact match to cron.
   5. Tree ~ -o myfolders.txt : sends output to myfolders.txt
   6. Tree ~ –P ‘\*’ –o myfiles.txt -s –D sends listing of all files in user account to a file. Include file size and date of last modification. Do info tree!
5. **Search disk for a file name**: Alternative to tree: find ~ -name <filename> (Do info find). Use “–iname” to ignore case. "~" denotes in user account, where "/" denotes entire disk.
6. Installing a package written for Python 2.7 in Python 3.4. See example cited below for adafruit/Adafruit Python ADS1x15 package.
7. Testing Internet connectivity with a Python script. Use my Python program *testinet.py* to test connectivity to Google network. Results logged to screen and the file: inettest.txt. Data logged on one line per test: date/time/message/number of tests made/number of tests failed.
8. Measure CPU temperature. CLI command: *vcgencmd measure\_temp*
9. Display memory usage: CLI command *free -m*
10. Utility to monitor stuff (possibly complicated) *RPImonitor* by ZavierBerger.
11. Simplified scheduling with "*at*". (sudo apt-get update && sudo apt-get install at) easy to use and dependable, runs from the cron utility in the background.
12. **Compare files for content differences**: *diff* utility is standard. To show two files side-by-side: *diff -y <file1> <file2>* *man diff* for more info. Google up [Best file comparison and difference tools for Linux.](https://www.tecmint.com/best-linux-file-diff-tools-comparison/)

# Packages I installed (name, source, special instructions)

1. Weather forecast data
2. ADC module for ADS1115 16-bit 4-channel ADC with Prog. Gain Amps. From Adafruit Learn topic [Raspberry Pi Analog to Digital Converters by Tony DiCola](https://learn.adafruit.com/raspberry-pi-analog-to-digital-converters/mcp3008). Installed from source on GitHub at [adafruit/Adafruit\_Python\_ADS1x15](https://github.com/adafruit/Adafruit_Python_ADS1x15). At time of install the instructions specified python which install the package for Python 2.7, modified instructions below to install for Python 3.4.
   * + 1. sudo apt-get update
       2. sudo apt-get install build-essential python-dev python-smbus git
       3. cd ~
       4. git clone https://github.com/adafruit/Adafruit\_Python\_ADS1x15.git
       5. cd Adafruit\_Python\_ADS1x15
       6. sudo python setup.py install <= Replace 'python' with 'python3' here.
   1. Per Lutz *Learning Python* p 681-682, verify with Python 3.4 interactive commands: >>> import sys and >>>sys.path to list the current path.
   2. While the simple change in instructions above seems to work, this is a learning opportunity creating setup.py scripts and for how to define PYTHONPATH per Lutz *Learning Python* Chapter 22 "Module Search Path" & Appendix A section on "Configuring Python". Also web resources (for starters search "Editing PYTHONPATH" in Raspberrypi.org, "How to permanently set environmental variables" and "Permanent Environment Variable for all users" in stackexchange)
3. GUI on Linux - 'mc' (installed on Zero2)
4. DropBox Uploader on RPi3 and RPZero. See "Saving Data in the Cloud" above.
5. Matplotlib 1.5.3 from [matplotlib.org](http://matplotlib.org/). On 24Dec16, Matplotlib is on RPi3B and Pavilion.
6. PIXEL fromRaspberryPi.org - hold off on other PIs! There are questions asked without much guidance for what is needed.
7. On RPi 2b: GrovePi+ software using instructions on www.dexterindustries.com, [Install Raspbian for Robots](https://www.dexterindustries.com/howto/install-raspbian-for-robots-image-on-an-sd-card/) . RPi now shows as 'dex' on internet connection
8. On RPi 2b, 3b: *InitialState* web streaming app. using instructions on [Raspberry Pi: Home Environment Hub](http://blog.initialstate.com/pi-home-environment-hub/#!prettyPhoto). Access key is stored in file APIcode.py file in Documents/PythonProjects/HTT. APIcode.py also has the access key for www.forecastio.com.
9. "[Requests](http://requests.readthedocs.io/en/master/)" to simplify use of Internet, replaces urllib which seems to have changed between Python versions.

# Physical interface APIs

**GPIO ZERO**

Introduction: <https://www.raspberrypi.org/blog/gpio-zero-a-friendly-python-api-for-physical-computing/>. Note this describes a number of input devices but nothing about use of DHT11/22 sensors. For this information look here: <http://docs.gadgetkeeper.com/pages/viewpage.action?pageId=7700673> or here:

http://www.instructables.com/id/Build-Your-First-IOT-with-a-Raspberry-Pi-DHT11-sen/

**GrovePi+** - See special section below.

**Raspberry Pi information -** <http://raspberry-pi-guide.readthedocs.io/en/latest/system.html>**. Visit this link to get information about the cpu, memory, system (host ame, ostype, osrelease, pidmax and poweroff\_cmd. Also Display usb devices in tree format with lsusb -t. Also, see mount points for disk devices.**

**SERIAL COMMUNICATION –** To enable communication between RPi and Arduino over USB ports. Hardware cable is USB-Serial conversion cable with Ethernet Cat-5 extension purchased from Adafruit.

Raspian software tools for this area of endeavor:

pySerial library. Search Raspberry Pi website. This is included in distro, verify with *aptitude*.

‘screen’ terminal emulator (?). Do *man screen* for manual pages

*lsusb* to list usb interfaces. In my specific case, I get 6 Devices listed with Arduino info shown as: “Bus 001 Device 005: ID 2341:0043 Arduino SA Uno R3 (CDC ACM)”.

*lsusb -d 0x2341:* with ending ‘:’, shows devices with vendor -0x2341 (code for Arduino)

Tree format for all usb devices is from *lsusb -t* command.

Verbose info on a usb device, do *lsusb -v -s5* | more for this case. For the USB cable, the Device Descriptor shows idVendor is Arduino, and Interface Descriptor.bInterfaceProtocol is *AT-commands*. (do man lsusb). Note the device number (005) will change if the cable is disconnected and reconnected.

*dmesg* (Display Message or Driver Message) Default action is to display all messages from the kernel ring buffer. Output format: [time from boot time] device name: message. Pipe to grep to sort through data (example: dmesg | grep -I usb). For more info, see <http://www.linuxnix.com/what-is-linuxunix-dmesg-command-and-how-to-use-it/> or Google dmesg

THIS WORKS TO IDENTIFY USB PORT  
*dmesg | grep -I tty* it returns [….] cdc-acm 1-1.4:1.0: ttyACM1: USB ACM device

*ls /dev/tty\**

General note dated 2011:

tty is the terminal associated with the current process  
tty1-tty63 are the virtual consoles (most systems only use the first 6 for text consoles though)  
tty0 is the currently selected virtual console  
ttyAMA0 is the built in UART on the Pi   
ttyprintk is a driver that allows output of messages through the system used by printk messages from the kernel.  
  
USB ports themselves don't have tty devices associated. However USB to serial adaptors and devices that contain or emulate USB to serial adaptors will have a tty device, Usually the name is ttyUSB<number> but some devices use other names.

References:

* <https://www.raspberrypi.org/forums/viewtopic.php?f=36&t=46321>
* Sample programs in Adafruit, Raspberry Pi, pySerial websites.

**Demonstration code:**

* ./PythonProjects/serialusb.py
* ./PythonProjects/HTT/HTT\_Unified.py

**Programmatic Shutdown of RPi initiated by pushbutton**

Shutdown is accomplished by scheduling a program that watches for a switch closure. Two methods here:

Common approach – on RPi startup, use /etc/rc.local to start a shutdown switch monitor script. This uses the RPi.GPIO library.

*rc.local script segment..*

*## Add launch script to /etc/rc.local. Ensure line ends with 'space'& to run in background.*

*sh /home/pi/Documents/PythonProjects/launcher.sh &*

*Launcher.sh*

#!/bin/sh

#launcher.sh

#Navigate to home directory, then to this directory, then execute python script.

cd /

cd home/pi/Documents/PythonProjects

sudo python GPIOIn.py

*GPIOIn.py* Python script to monitor switch and make OS shutdown call.

import RPi.GPIO as GPIO

import time

from subprocess import call

import sys

def test(ipin):

sw\_pin = ipin

GPIO.setmode(GPIO.BCM) # Board pin numbering used.

GPIO.setup(sw\_pin, GPIO.IN) # Define a pin as input.

return (GPIO.input(sw\_pin)!=0)

sw\_pin = 5

GPIO.setmode(GPIO.BCM) # Processor pin numbering used.

GPIO.setup(sw\_pin, GPIO.IN) # Define a pin as input.

r = 3

for l in range(r):

input\_value = GPIO.input(sw\_pin)

print( str(r-l),' Pin status: ', input\_value)

time.sleep(0.5)

print("Push the button")

while True:

if test(sw\_pin):

print ".",

time.sleep(0.1)

else:

# Pushbutton press will shutdown. No keyboard needed.

call(["shutdown","-h","now"])

Second approach to programmatic shutdown with pushbutton actuation:

This uses a breakout board made by LowPowerLab.com called ATXRaspi.

<https://lowpowerlab.com/guide/atxraspi/>

# Python Notes

References:

1. Ref A: Python 3.4 Tutorial <https://docs.python.org/3.4/tutorial/index.html>
2. Ref B: Python Tutorial: <http://www.tutorialspoint.com/python/index.htm>
3. Ref C: Python.org <https://www.python.org/doc/>
4. Ref D: Instructables [Raspberry Pi Class](http://www.instructables.com/class/Raspberry-Pi-Class/?utm_source=classesNL&utm_medium=email) with push\_reset
5. How to see what names are defined by a [module](https://docs.python.org/3.4/tutorial/modules.html#modules)? In the Python interpreter, type 'dir(modulename) to get a list of names in the module (properties and methods).
6. What names are defined currently in the interpreter? Type 'dir()'.
7. How to get documentation for names? dir(modulename) yields properties and methods. help(modulename) yields description of names, help(modulename.name) yields brief help for name.
8. To list names of built-in functions and variables, import the module builtins and type dir(builtins)
9. To end a long listing in Python CLI, do 'Q' (not case sensitive). To end listing and exit Python, do Ctrl-Z.
10. Python interactive command history is stored in user directory in file .python\_history. (Verify this)
11. GUI for my applications: use tkinter package. Introduced in Programming in Raspberry Pi (Monk). Use dir(Tk) and help(Tk) or help(Tk.<name>) or help(Wm) to find out details, as Monk's book is short on explanation. Note: Class Wm in tkinter, provides functions for communication with the window manager.
12. Defining long strings: use triple quotes ''' or """. Long strings can span multiple lines and can include special characters (NEWLINEs, TAB, etc.) Ref B:
13. Learning module and class methods and data? ans: Google search *'How how I look inside a Python object?*' and going to StackOverflow.com: Working in Python, first import the module, then

in Python command line (or shell), type: dir(functionname)

OR help(module or module.class)

OR: Python has a strong set of introspection features.

Take a look at the following [built-in functions](http://docs.python.org/library/functions.html):

* type()
* dir()
* id()
* getattr()
* hasattr()
* globals()
* locals()
* callable()

type() and dir() are particularly useful for inspecting the type of an object and its set of attributes, respectively.

OR: or object.\_\_dict\_\_

OR: to see actual source code:

>>> import string

>>> string.\_\_file\_\_

'/usr/lib/python2.5/string.pyc'

The \*.pyc file is compiled, so remove the trailing 'c' and open up the uncompiled \*.py file in your favorite editor or file viewer:

/usr/lib/python2.5/string.py

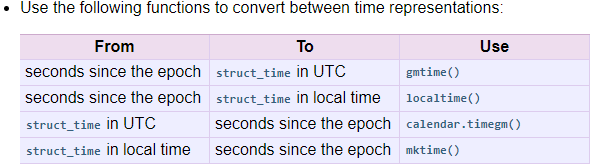
I've found this incredibly useful for discovering things like which exceptions are raised from a given API. This kind of detail is rarely well-documented in the Python world.

1. **Programmatic access to system information using standard library modules**. Import ‘os’ and see methods described here: <http://raspberry-pi-guide.readthedocs.io/en/latest/programming.html>
2. Networking in Python.

Learning Python with Raspberry PI, Networked Python. Program code can be downloaded from <www.wiley.com/go/pythonraspi>

What is my IP address? Ans. LXTerminal command ifconfig. Or in GUI, hover mouse over the network icon in upper right corner. Another alternate: use iPad FING app!

1. Sending images to the cloud. See above "Saving data to the cloud" for Dropbox and Adafruit IO examples.
2. Time Objects. In Python: *import time* and do *help(time)*
3. time.time() returns the time in seconds since the epoch as a floating point number. Some systems return time with fractional seconds truncated (precision to nearest second). Epoch is system defined, in UNIX it is generally January 1st, 1970.
4. Another standard representation of *time* is a tuple of 9 integers representing number of seconds since the Epoch. Do help(time) for more information.
   1. For GMT, the actual value may be retrieved by calling gmtime(). Do print(time.gmtime()) to show the tuple for GMT time.
   2. For local time, the actual value may be retrieved by calling localtime(). Do print(time.localtime()) to show the tuple for local time.
5. Yet another representation of time is as an ASCII string. It is derived from one of the time tuples. Day-name Mon Day-of-Mon HH:MM:SS Year. Do help(time.asctime()). If time tuple not specified, gets time from time.localtime() which converts seconds since start of epoch to local time tuple. If seconds not specified, gets seconds from local current time. Print(now()) is equivalent to print(time.asctime(time.localtime())).
6. print(*time.time())* shows local time in seconds since Epoch.
7. Print(*time.strftime(format,t*) Convert a tuple of struct-time representing a time as returned by *gmtime()* or *localtime()* to a string as specified by the format argument. If *t* is not provided, the current time is returned by *localtime()* is used. For format directives, reference:<https://docs.python.org/2/library/time.html#time.strftime>
8. Reference for time class: <https://docs.python.org/2/library/time.html>
9. Reference for struct\_time variables: <https://docs.python.org/2/library/time.html#time.struct_time>
10. Table of methods to convert between struct\_time and “seconds since the epoch”:



1. Reference my program: TimeDemo.py
2. [Python Standard Library](https://docs.python.org/3.4/contents.html) is a good on-line reference for all recent versions of the language. See [shutil](https://docs.python.org/3.4/library/shutil.html#directory-and-files-operations) for example.
3. Execute a BASH script with argument from Python program. Google search "[executing shell script with argument from Python](http://stackoverflow.com/questions/19325761/python-executing-shell-script-with-argumentsvariable-but-argument-is-not-rea)". This code works:

# CallBASH.py

""" Demonstration of calling a bash script from Python

"""

from subprocess import Popen

localFile = 'today.csv'

remoteFile = localFile

Process=Popen('~/Dropbox-Uploader/dropbox\_uploader.sh upload %s ./Apps/%s' % (str(localFile), str(localFile)), shell=True)

# Git Notes

See my separate document on this topic entitled Git-WorkFlowForNewRepository.docx

# GrovePi+ Notes

GrovePi installation requires a special version of Raspian. Instructions for loading the new OS are found on [Dexter Industries](https://www.dexterindustries.com/howto/install-raspbian-for-robots-image-on-an-sd-card/) [website](https://www.dexterindustries.com/howto/install-raspbian-for-robots-image-on-an-sd-card/).

Remote login: Use Putty with the unique host name and password for GrovePi.

1. Host name: dex.local
2. username pi
3. password: robots1234

File transfer using FileZilla: Need the IP address for the GrovePi. I used Fing, this is also on the Pi GUI screen, upper right. Username: pi, password: robots1234, port 22 (the default for SFTP)

High Tech Thermometer application (HTThermometer.py)

The GrovePi comes with a thermistor sensor which is not very accurate. Use instead a DHT22 package (I have one) and follow the example on page 24, GroveP GettingStarted booklet (Home Weather Display). For concept development, I put the DHT22 in a solderless breadboard and made a cable to go from GrovePi to the breadboard. Plug in the 'yellow' signal line to DHT22 signal pin.

Issues to solve on my application, HTThermometer.py.

* *~~import urllib.request~~* ~~does not work on RPi, only on Windows Python. Find equivalent.~~
* HTThermometer.py with getForecast only works in Python3 (not 2) but
* HTThermometer\_HT22 only works in Python2 (not 3). Try to install GrovePi for Python3.
* Polish needed: format floats in strings to have fixed number of decimals.
* Place strings in LCD without overwriting. See example: */home/pi/Desktop/GrovePi/Software/Python/grove\_rgb\_lcd/example3.py* that uses *setText\_noRefresh()*