Setting up a Raspbian Linux System on an SD Card

How the initial step is done depends on the type of computer system you have that connects to the Internet. The downloading instructions at and compressed system files are available from

http://raspberrypi.org

where you go to Downloads tab, scroll down to RASPBIAN and select "Download ZIP".

The current system labeled "2015-11-21-raspbian-jessie.zip" has about 1.3 GBytes (about 4.0 GB when unzipped).
Read the "Image Installation Guides" link in the RASPBIAN section of the download page to see how to proceed when downloading into a Linux, Windows, or Mac system. The following details are how I proceeded when downloading into a working Raspberry Pi.

The Raspberry Pi browser downloaded it into the directory /home/pi, the home directory of the pi user. You then need to unzip it:

unzip 2015-11-21-raspbian-jessie.zip

This will require about 10 min on the Raspberry Pi 2.

The next step using the “dd” instruction must be done carefully or we risk wiping out the working system. We first check what filesystems are on the working system by doing:

df -h

which might show something like the following:

```
Filesystem      Size  Used Avail Use% Mounted on
rootfs           30G   14G   15G  47% /
/dev/root        30G   14G   15G  47% /
devtmpfs        428M     0  428M   0% /dev
tmpfs            87M  284K   87M   1% /run
tmpfs           5.0M     0  5.0M   0% /run/lock
tmpfs           173M     0  173M   0% /run/shm
/dev/mmcblk0p1   56M   20M   37M  35% /boot
```

Next I put new SanDisk 32 GB micro SD card into USB adapter and plugged into open USB slot and checked the filesystems again:

df -h

now had a new line of output at its end:

```
/dev/sda1        30G   32K   30G   1% /media/6661-3738
```

This tells me to use /dev/sda1` in the next instruction:

umount /dev/sda1

df -h should no longer show the /dev/sda1 line.

Next you need to copy the entire system with its two partitions onto you new SD card.
Assuming that your test above showed that the USB adapter is in /dev/sda1, you will use /dev/sda, not /dev/sda1 in the following instruction. Otherwise, you will need to used whatever it showed minus the 1 at the end.

sudo dd bs=4M if=2015-11-21-raspbian-jessie.img of=/dev/sda

This took 4 minutes and 40 seconds to transfer the system image to the SD card.
(The sudo may not have been necessary since the USB ports are owned by the pi user.)

As a precaution to be sure that all has been written to /dev/sda, do the following command

sync

Now, if you unplug the USB adapter and plug it back in, the system should recognize its new partitions. Doing

df -h

should now show lines like the following two at the end of its listing:

```
/dev/sda1        56M   19M   37M  34% /media/boot
/dev/sda2       2.9G  2.3G  435M  85% /media/13d368bf-6dbf-4751-8ba1-88bed06bef77
```

Don't worry that the space used is less than the capacity of the SD card. This will soon be fixed.

To use your new card, shutdown the Pi by typing

sudo halt

and when the lights stop blinking after a few seconds, unplug its power.

Place your newly filled micro SD card into the SD card slot on the Pi and boot it up with a keyboard and video connection.

On your first boot with the new system, a desktop will appear.

Using the menu, you should select Menu/Preferences/Raspberry Pi Configuration
and then set the following:
Set Localization/Location to en (English), US (USA), UTF-8
Set Localization/Time Zone to US, Pacific-New
Set Localization/Keyboard to United States, English (US, international)
Set System/Filesystem Expand Filesystem

A reboot will then be necessary.

If the mouse response is slow, adding **usbhid.mousepoll=0** to the end to the boot command line as follows should solve the problem:

The original **cmdline.txt** is shown by doing:

```
$ cat /boot/cmdline.txt
dwc_otg.lpm_enable=0 console=ttyAMA0,115200 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline fsck.repair=yes rootwait
```

Edit it with the nano editor

$ **sudo nano /boot/cmdline.txt**

to add a space and **usbhid.mousepoll=0** at its end of the same line so it looks like

$ **cat /boot/cmdline.txt**
dwc_otg.lpm_enable=0 console=ttyAMA0,115200 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline fsck.repair=yes rootwait usbhid.mousepoll=0

Be sure you have made the change correctly, save it, and then reboot for this change to take effect.

$ **sudo reboot**

At this point the default user name is "pi" and the default password is "raspberry".

When it boots up again, you can check if you have a network connection by doing

ifconfig

If you only are using an Ethernet connection, this should show something like

```
eth0      Link encap:Ethernet  HWaddr b8:27:eb:83:ca:91
          inet addr:192.168.2.11  Bcast:192.168.2.255  Mask:255.255.255.224
          inet6 addr: fe80::ba27:ebff:fe83:ca91/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:378 errors:0 dropped:0 overruns:0 frame:0
          TX packets:375 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:113118 (110.4 KiB)  TX bytes:44493 (43.4 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:12 errors:0 dropped:0 overruns:0 frame:0
          TX packets:12 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1344 (1.3 KiB)  TX bytes:1344 (1.3 KiB)
```

To check your connection, you can try to ping yahoo.com by doing (You will need to stop it by doing a entering Ctrl-C.)

ping yahoo.com

In another installation, I used a Edimax WiFi connection instead of an Ethernet connection.

In that case, I needed to edit /etc/wpa_supplicant/wpa_supplicant.conf to add a network clause after the first two lines in that file. Editing can be done by doing

sudo nano /etc/wpa_supplicant/wpa_supplicant.conf

The network clause provides the SSID and PSK for the sending network.
For example, if I do this at home, it needs to look like:

```
network={
   ssid="Yosemite_Foothills"
   psk="TVAdsDriveMeCrazy"
}
```

After a reboot, it should be able to connect to the WiFi network with the specified ssid and psk.
Most school and corporate connections are likely to require more items in the network clause.

To make the command-line console use a larger font, I like to do the following:

sudo nano /etc/default/console-setup

and change the FONTFACE and FONTSIZE lines to

FONTFACE="TerminusBold"
FONTSIZE="22x11"

Ctrl-O saves the file and Ctrl-X exit the nano editor.

This change will take place after the next reboot.

If all is well, we should update the package database by doing:

sudo apt-get update

When that finishes, we should ask it to upgrade any packages to newer versions by doing:

sudo apt-get dist-upgrade

Choose "yes" when it asks if you want to upgrade.

You will occasionally want also upgrade the system firmware by doing

sudo rpi-update

but there is a slight chance that this may cause the system to have difficulties if the latest rpi-update has a problem.

After doing rpi-update, it is necessary to once again perform a reboot for it to take effect.

When I added the Ethernet connection to the WiFi-connected system, I then had both connections operating.

At this point, the file **/etc/network/interfaces** contained the following:

```
auto lo
iface lo inet loopback

auto eth0
allow-hotplug eth0
iface eth0 inet manual

auto wlan0
allow-hotplug wlan0
iface wlan0 inet manual
wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf

auto wlan1
allow-hotplug wlan1
iface wlan1 inet manual
wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
```

which is the how it was installed. We did not need to edit it.

**** Setting up a server for remote desktop viewing ****

Virtual network computing (vnc) is useful for sharing a desktop or simply using a computer desktop from a remote computer. Its server is installed by doing:

sudo apt-get install tightvncserver

Before you start the vncserver, you need to have the X-Window system operating. You may have already told the system to automatically start the X-Window system by a setting in raspi-config, but if not, it can be started by using the command

startx

(Ignore the error message “FATAL: Module g2d_23 not found.”)
The startx command can be done from a remote system via ssh.

Then you start the vnc server by doing

vncserver

The first time it is started, it will ask for a password that needs to be a maximum of 8 characters.
Let's just use "**password**" for now. It will also ask if you want to specify a view-only password, that is a password that only allows viewing of the desktop, but not mouse or keyboard interaction. Say “yes” and enter the view-only password. Let's use “letmein” for the view-only password.

On a remote Raspberry Pi or other Linux system, one can install a viewer called "**xtightvncviewer**" or run a program the might be called "**Remmina Remote Desktop**" on the program menu. On a Windows system, the choice might be called "**Real VNC**".

sudo apt-get install xtightvncviewer

The run it on the remote system by typing

xtightvncviewer

Then enter the IP address of the server system followed by a colon and a display number (usually 1)
On my system, I entered "192.168.2.10:1". It then asked for the password of the server. In this case, it is just "password". The pi's desktop should then appear on the remote system.

This does not seem to work right with my Ubuntu system as a viewer. I'll need to figure it out later.

The suggested procedure is at

https://www.raspberrypi.org/documentation/remote-access/vnc/

which I think I followed, but did not want it to make it start at boot. I prefer to manually start it by doing

vncserver

as above. Stopping the vncserver that is using display :1 is done by doing

vncserver -kill :1

Adding Some Useful Software

**** vim ****

I like to use an editor called vim that is particularly useful for programming. To install it I do

sudo apt-get install vim

When it asks if I want to proceed, I press Enter to select the capitalized "Y" option for "yes".

We should edit a configuration file for vim to have it always use syntax highlighting, work with a black background, and return to the last line edited when reopened. These modifications are done by doing

sudo vim /etc/vim/vimrc

and moving to each of the following lines and removing # symbol (curser down to it and press del key) at the start of the line.
If you press other keys, strange things might happen and you should start over by pressing the escape key followed by ":q!" (without

quotes). You will need to learn more about the editor later.

#syntax on

    becomes

syntax on

#set background=dark

    becomes

set background=dark

#if has("autocmd")
au BufReadPost * if line("'\"") > 1 && line("'\"") <= line("$") | exe "normal! g'\"" | endif
#endif

    becomes

if has("autocmd")
** au BufReadPost * if line("'\"") > 1 && line("'\"") <= line("$") | exe "normal! g'\"" | endif**
endif

To finish, enter ":wq" (without the quotes).

**** Python Imaging Library for Python2.7 and Python3 ****

I'm not sure the next two commands were necessary, but they probably were.

sudo apt-get install python-pip
sudo apt-get install libtiff5-dev libjpeg-dev zlib1g-dev libfreetype6-dev liblcms2-dev libwebp-dev tcl8.5-dev tk8.5-dev python-tk
sudo apt-get install python-dev

Downloaded Pillow-3.0.0.tar.gz from https://pypi.python.org/pypi/Pillow and then unzip and untar it:

gunzip Pillow-3.0.0.tar.gz
tar xf Pillow-3.0.0.tar

Untarring created a directory which you need to change into making it your current working directory:

cd Pillow-3.0.0

The next step creates and installs the required graphics programming libraries:

sudo python setup.py install

It is necessary to exit and reopen the command line window for the Pillow packages to be
found by the python programming environment.

In the new window enter

python
>>> **from PIL import Image**
>>>
(Use ctrl-D to exit the python interpreter.)

and see that it now exists in the system by not giving any errors.

We need to follow a similar process for the Python 3 version
 (python3-setuptools and python3-dev are already installed):

cd Pillow-3.0.0
sudo python3 setup.py install

After once again exiting the command-line window and reopening it, check that PIL works on both versions of Python:

python3
>>> **from PIL import Image**
>>>
(Use ctrl-D to exit the python interpreter.)

and see that it now exists in the system by not giving any errors.

The Pillow-3.0.0.tar file and Pillow-3.0.0 directory tree are no longer needed and can be deleted:

rm Pillow-3.0.0.tar
rm -rf Pillow-3.0.0

**** python-imaging-doc **** recommended documentation for python imaging library

sudo apt-get install python-imaging-doc
sudo apt-get install python-imaging-doc-html

**** python-scipy ***** Scientific subroutine programs for python (numpy is already installed in Raspbian system)

sudo apt-get install python-scipy
sudo apt-get install python3-scipy

**** locate **** Program for accessing database of file locations on computer

sudo apt-get install locate
sudo updatedb

The command "sudo updatedb" should be done whenever you want to be sure that the database
of the file locate program has the newest changes in the file system.

15 identical error messages were shown:
/usr/bin/find: `/run/user/1000/gvfs': Permission denied

**** gifsicle **** Programs for viewing and creating animated gifs

sudo apt-get install gifsicle

**** feh **** Convenient image display program

sudo apt-get install feh

**** gimp **** A powerful photo editing program

sudo apt-get install gimp gimp-help-en gimp-data-extras

**** inkscape **** A vector graphics editing program

sudo apt-get install inkscape

**** bkchem **** A program for drawing chemical structures

sudo apt-get install bkchem

**** avogadro **** A program for creating and viewing chemical molecular structures in 3D

sudo apt-get install avogadro

The next step is unnecessary in the new Jessie Raspbian system. It is automatically in the system or is installed with avogadro.
In that case, avogadro will work right away. Earlier versions required this installation of libgl1-mesa-dri:
sudo apt-get install libgl1-mesa-dri

**** gelemental **** A program providing an interactive Periodic Table of Elements

sudo apt-get install gelemental

**** evince **** PDF file viewer

sudo apt-get install evince

Setting up Programming Tools and Installing Example Programs

Special Note: The python program I wrote to upload microprocessor instructions into a ATmega328P needed modification for use with Python3 and also a different modification for use with the new Jessie Raspbian system. The old Wheezy Raspbian system used the spi-bcm2708 kernel driver and Jessie uses spi-bcm2835. The new driver takes control of the chip enable pin CE0 (GPIO 7, pin 24 on the header) whereas it appears the old one did not. When uploading programs to the ATmega328P, the ATmega328P reset line doubles as a program enable line and needs to be held down during the entire program upload. The new driver, however, toggles CE0 for each transfer. The solution is connect the ATmega328P reset to CE1 (GPIO 7, pin 26 on the header) and keep it low during the entire upload process. This works with both drivers. The modified uploading programs are at

`http://yosemitefoothills.com/RaspberryPiWorkshopNotes.html`

Programs that use the SPI system to do normal SPI transfers instead of ATmega328P programming, need to use CE0 (GPIO 8, pin 24 on the header). This includes my python programs `SPITesting.py`, `VoltmeterTest.py`, and `Logging/spiReading.py`.

I have created a compressed tar file named **`Documentation_GeigerCounter_Music_Pictures_Programming_Sounds.tgz`** which contains a bunch of useful stuff. If it is unzipped and untarred into the /home/pi directory, some of the following steps will be unnecessary.

**** Setting up Programming Directories *****

```
mkdir Programming
cd Programming
mkdir C
mkdir C++
mkdir Python
mkdir Java
```

Copied various programs into those directories.

The java and javac are already included in the Raspbian system.

A java example named Hi2 uses sounds. To get the sounds to go down the HDMI video cable, I needed to edit **/boot/config.txt** by doing

sudo nano /boot/config.txt

and change

#hdmi_drive=2

to

hdmi_drive=2

A reboot is necessary for this to take effect.

This lets the sound to to the HDMI connector where my converter box can separate it out or where it can be presented by a sound-capable HDMI monitor/TV.

**** Assembly programming for ATmega328P processor ****

You might want to look at the following tutorial:

http://www.instructables.com/id/Command-Line-Assembly-Language-Programming-for-Ard/

There is an avra assembler available using apt-get install, but it is version 1.2.3 Build 1 (15. November 2007) so we will install a newer version which we must compile from its source code.

From the following site download **avra-1.3.0.tar.bz2**:

http://sourceforge.net/projects/avra/

Make a suitable directory and make it the current working directory:

```
mkdir Programming
cd Programming
mkdir Assembly
cd Assembly
```

and put avra-1.3.0.tar.bz2 into that directory.

```
mv /home/pi/avra-1.3.0.tar.bz2 /home/pi/Programming/Assembly
```

Next bunzip2 it:

```
bunzip2 avra-1.3.0.tar.bz2
```

and untar it:

```
tar xf avra-1.3.0.tar
```

and enter the newly-created directory:

```
cd avra-1.3.0
```

Compile and install the avra assembler:

```
sudo apt-get install automake
cd src
touch NEWS
touch ChangeLog
mv ../AUTHORS .
mv ../README .
Aclocal
  (Got warning:aclocal: warning: autoconf input should be named 'configure.ac', not 'configure.in'
  so I did cp configure.in configure.ac)
autoconf
  (Then I got the warning: autoconf: warning: both `configure.ac' and `configure.in' are present.
  autoconf: warning: proceeding with `configure.ac'.)
automake -a
  (And then got warning: automake: warning: autoconf input should be named 'configure.ac', not 'configure.in'
  automake: warning: 'configure.ac' and 'configure.in' both present.
  automake: proceeding with 'configure.ac'
  configure.ac:3: warning: AM_INIT_AUTOMAKE: two- and three-arguments forms are deprecated.  For more info, see:
  configure.ac:3: http://www.gnu.org/software/automake/manual/automake.html#Modernize-AM_005fINIT_005fAUTOMAKE-invocation
  configure.ac:6: installing './compile'
  configure.ac:3: installing './install-sh'
  configure.ac:3: installing './missing'
  Makefile.am: installing './INSTALL'
  Makefile.am: installing './COPYING' using GNU General Public License v3 file
  Makefile.am:     Consider adding the COPYING file to the version control system
  Makefile.am:     for your code, to avoid questions about which license your project uses
  Makefile.am: installing './depcomp'
  automake: warning: autoconf input should be named 'configure.ac', not 'configure.in'
  automake: warning: 'configure.ac' and 'configure.in' both present.
  automake: proceeding with 'configure.ac')
./configure
  (Got some more warnings:
  makeCDPATH="${ZSH_VERSION+.}:" && cd . && /bin/bash /home/pi/Programming/Assembly/Avra_Compiler/avra-1.3.0/src/missing aclocal-1.14
  aclocal-1.14: warning: autoconf input should be named 'configure.ac', not 'configure.in'
  aclocal-1.14: warning: 'configure.ac' and 'configure.in' both present.
  aclocal-1.14: proceeding with 'configure.ac'
   cd . && /bin/bash /home/pi/Programming/Assembly/Avra_Compiler/avra-1.3.0/src/missing automake-1.14 --gnu
  automake-1.14: warning: autoconf input should be named 'configure.ac', not 'configure.in'
  automake-1.14: warning: 'configure.ac' and 'configure.in' both present.
  automake-1.14: proceeding with 'configure.ac'
  configure.ac:3: warning: AM_INIT_AUTOMAKE: two- and three-arguments forms are deprecated.  For more info, see:
  configure.ac:3: http://www.gnu.org/software/automake/manual/automake.html#Modernize-AM_005fINIT_005fAUTOMAKE-invocation
  automake-1.14: warning: autoconf input should be named 'configure.ac', not 'configure.in'
  automake-1.14: warning: 'configure.ac' and 'configure.in' both present.
  automake-1.14: proceeding with 'configure.ac'
  CDPATH="${ZSH_VERSION+.}:" && cd . && /bin/bash /home/pi/Programming/Assembly/Avra_Compiler/avra-1.3.0/src/missing autoconf
  autoconf: warning: both `configure.ac' and `configure.in' are present.
  autoconf: warning: proceeding with `configure.ac'.
  …)

sudo install avra /usr/bin
```

(The last command was used instead of 'sudo make install' because 'sudo make install' puts
the file in '/usr/local/bin' instead of '/usr/bin'. This was easier than changing 'Makefile'.)

The success of this process can be tested by doing:

avra

and seeing it produces its version and help information.

The avra-1.3.0.tar file and avra-1.3.0 directory tree are no longer needed and can now be deleted:

```
cd /home/pi/Programming/Assembly
rm avra-1.3.0.tar
rm -rf avra-1.3.0
```

We still need an include file '**m328Pdef.inc**' which defines all the terms used to define the pins, ports, etc. of the particular processor for which the assembly is being done. This file can be copied from

https://github.com/DarkSector/AVR/blob/master/asm/include/m328Pdef.inc

You should place it where you have your assembly source file or somewhere it will be found by the assembler.

For now, put it into your '/home/pi/Programming/Assembly/' directory:

```
mv /home/pi/m328Pdef.inc /home/pi/Programming/Assembly/
```

In order to communicate to the Gertboard via an spi connection, we need a python module named **spidev**.
The process is similar to how we created the python imaging library modules. If that has been done the installations of python-dev and python3-dev are redundant. No harm trying if you are unsure.
The following instructions obtain the source code for that module, compile it, and install it::

```
cd /home/pi/Programming/Python
mkdir python-spi
cd python-spi
wget https://raw.github.com/doceme/py-spidev/master/setup.py
wget https://raw.github.com/doceme/py-spidev/master/spidev_module.c
touch README.md CHANGELOG.md
sudo apt-get install python-dev
sudo python setup.py install
sudo apt-get install python3-dev
sudo python3 setup.py install
```

The success of this process can be seen if the python and python3 programs are able to **import spidev** just as we tested the imaging libraries by doing **from PIL import Image**.

If all is well, the **python-spi** directory and its contents are no longer needed and can be deleted:

```
cd /home/pi/Programming/Python
sudo rm -rf python-spi
```

Setting Up a WiFi Access Point

I am following instructions at:

http://elinux.org/RPI-Wireless-Hotspot

sudo apt-get install hostapd udhcpd

Edited as superuser (i.e. using sudo) the file **/etc/default/udhcpd** to change line with

DHCPD_ENABLED="no"

to

#DHCPD_ENABLED="no"

Edited as superuser **/etc/udhcpd.conf** to contain (along with defaults):

```
start 192.168.42.2 # This is the range of IPs that the hostspot will give to client devices.
end 192.168.42.20
interface wlan0 # The device uDHCP listens on.
remaining yes
opt dns 192.168.2.3 # The DNS servers client devices will use.
opt subnet 255.255.255.0
opt router 192.168.42.1 # The Pi's IP address on wlan0 which we will set up shortly.
#opt    wins    192.168.10.10
#option dns     129.219.13.81   # appened to above DNS servers for a total of 3
opt lease 864000 # 10 day DHCP lease time in seconds
```

Edited as superuser **/etc/network/interfaces** to become:

```
auto lo
iface lo inet loopback

auto eth0
allow-hotplug eth0
iface eth0 inet manual

auto wlan0
iface wlan0 inet static
  address 192.168.42.1
  netmask 255.255.255.0
up iptables-restore < /etc/iptables.ipv4.nat
```

```
#auto wlan0
#allow-hotplug wlan0
#iface wlan0 inet manual
#wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf

#auto wlan1
#allow-hotplug wlan1
#iface wlan1 inet manual
#wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
```

Edited as superuser **/etc/default/hostapd** to specify location of configuration file changing

```
#DAEMON_CONF=""
```

to

```
DAEMON_CONF=/etc/hostapd/hostapd.conf
```

Used an editor as superuser to create **/etc/hostapd/hostapd.conf** and put the following lines into it:

```
interface=wlan0
driver=nl80211
ssid=ACM_Workshop
hw_mode=g
channel=6
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_passphrase=Lets_learn_some_stuff
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
```

The ssid and wpa_passphrase lines are, of course, your choice. Matching lines, however, must be in receiving systems.

See **https://w1.fi/cgit/hostap/plain/hostapd/hostapd.conf** for details of these options

On the receiving system, the file **/etc/wpa_supplicant/wpa_supplicant.conf** will need to have the following network clause added after its initial lines. The file, edited as superuser, contains:

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
network={
    ssid="ACM_Workshop"
    psk="Lets_learn_some_stuff"
}
```

Also on the receiving system, **/etc/network/interfaces** should include

```
(check this!!!)
auto wlan0
allow-hotplug wlan0
iface wlan0 inet manual
wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf
```

Tell the Linux kernel that it should allow forwarding of network traffic by doing:

```
sudo sh -c "echo 1 > /proc/sys/net/ipv4/ip_forward"
```

This takes effect immediately.

Edit **/etc/sysctl.conf** to remove the # from the start of the line

```
#net.ipv4.ip_forward=1
```

so that it becomes

```
net.ipv4.ip_forward=1
```

This causes IP forwarding to be enabled automatically every time at boot.

To enable NAT (network address translation) do:

```
sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
sudo iptables -A FORWARD -i eth0 -o wlan0 -m state --state RELATED,ESTABLISHED -j ACCEPT
sudo iptables -A FORWARD -i wlan0 -o eth0 -j ACCEPT
sudo sh -c "iptables-save > /etc/iptables.ipv4.nat"
```

To check that these iptables commands worked you can do

```
cat /etc/iptables.ipv4.nat
```

and you should see:

```
# Generated by iptables-save v1.4.14 on Sat May 23 22:15:58 2015
*filter
:INPUT ACCEPT [4:304]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [4:304]
```

```
-A FORWARD -i eth0 -o wlan0 -m state --state RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -i wlan0 -o eth0 -j ACCEPT
COMMIT
# Completed on Sat May 23 22:15:58 2015
# Generated by iptables-save v1.4.14 on Sat May 23 22:15:58 2015
*nat
:PREROUTING ACCEPT [0:0]
:INPUT ACCEPT [0:0]
:OUTPUT ACCEPT [4:304]
:POSTROUTING ACCEPT [0:0]
-A POSTROUTING -o eth0 -j MASQUERADE
COMMIT
# Completed on Sat May 23 22:15:58 2015
```

The following was recommend but appears to be unnecessary as the installation seemed to do this:
Just in case that is not the case, to make the access point start at boot do:

sudo update-rc.d hostapd enable
sudo update-rc.d udhcpd enable

********** Start of June 27, 2015 Update **********

There is a daemon that should be purged. It cause the WiFi to be messed with when an
Ethernet connection is made and is said to be poorly written:

sudo apt-get purge ifplugd

Also, make sure that /etc/network/interfaces for the interface being used on the wireless hosting (one with hostapd) system does not have both “auto wlan0” and “allow-hotplug wlan0”;
remove “allow-hotplug wlan0” and leave “auto wlan0”. The receiving system can have both entries.

We need to also install a “caching name server”, one that remembers can forward requests to an outside name server and can cache recent answers.

sudo apt-get install bind9 bind9-doc dnsutils

installs the “Berkeley Internet Name Daemon” (BIND) which defaults to the caching mode that we want. The daemon program name is named and has configuration files in /etc/bind9/. One can check the daemon's status by doing:

service bind9 status

and getting a reply

[ok] bind9 is running.

There is an associated program handling security matters called rndc that is also installed. Its status can be checked by doing

sudo rndc status

and getting a reply that is something like:

```
version: 9.8.4-rpz2+rl005.12-P1
CPUs found: 1
worker threads: 1
number of zones: 18
debug level: 0
xfers running: 0
xfers deferred: 0
soa queries in progress: 0
query logging is OFF
recursive clients: 0/0/1000
tcp clients: 0/100
server is up and running
```

To make it start up at boot time we also do:

sudo update-rc.d bind9 defaults

and get a reply:

update-rc.d: using dependency based boot sequencing

At this point, all WiFi connected computers can get name service and connect to the outside world if such a connection is plugged into the Ethernet port, but we still need to use IP addresses between within our local network. We could solve this by placing entries into /etc/hosts or by setting up the zone database for bind. I am not going to bother setting up the zone database, but here is an example of using /etc/hosts:

cat /etc/hosts

```
127.0.0.1 localhost
::1             localhost ip6-localhost ip6-loopback
fe00::0         ip6-localnet
ff00::0         ip6-mcastprefix
ff02::1         ip6-allnodes
ff02::2         ip6-allrouters

127.0.1.1          raspberrypi-server
192.168.42.6       raspberrypi4
192.168.42.4       raspberrypi1
```

You will want to edit the last few lines to suit your needs.

Changing hostname and password and Enabling SPI and SSH Communication

Using the menu, you should select Menu/Preferences/Raspberry Pi Configuration
and then change your hostname and password. (**raspberrypi-server** and **ACMWorkPiB208** for this workshop.)
SPI will need to be enabled. SSH should already be enabled.
If you have a camera, it needs to be enabled here also.

You will need to reboot for these changes to take effect.

I choose a pretty good password, because I will be connected to the school network and
want to make it more difficult to break.

The SSH protocol is used to operate a system from a remote computer so that no keyboard or terminal is needed.
The SPI feature will be used to send programs to the ATmega328P processor on the Gertboard.

Installing the Apache Web Server with PHP Programming

sudo apt-get install apache2
sudo apt-get install php5 libapache2-mod-php5

Browsing to http://<address of the server> will now show the Apache Debian default page with a line
stating "It Works!".
The source for that page is at /var/www/html/index.html. You can then place you web pages in the
/var/www/html/ directory but you must do it as the superuser root. This is usually done by doing

cd /var/www/html/
sudo vi index.html
and making subdirectories by doing
sudo mkdir <name of subdirectory>

php code is inserted into normal web pages by starting with
<?php
and ending with
?>
but it is necessary then to change the ending of the page from .html to .php

See various sources on the web and in books for how to write html and php code.