Setting up a Raspbian Linux System with WiFi Access Point, Apache Server, and Example Programs

Getting Started

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 "2016-05-10-raspbian-jessie.zip" has about 1.4 GBytes (about 3.8 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 following are commands that must be entered in a command-line window or on a console terminal.

(If this is done on another system, the details like /dev/sd* or /dev/mmcblk** will be different.)

The Raspberry Pi browser downloads it into the directory /home/pi/Downloads/ . Opening a command-line window will place you at your home directory /home/pi/ so you need to change to the Downloads directory to work with it once it has finished downloading (about 1 hour required).

`cd /home/pi/Downloads`

I then need to unzip it:

`unzip 2016-05-10-raspbian-jessie.zip`

This will require about 10 min on the Raspberry Pi 2 and will create an image file:

`2016-05-10-raspbian-jessie.img`

The next step using the “dd” instruction must be done carefully or we risk wiping out the working system. I first check what filesystems are mounted 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 a new SanDisk 32 GB micro SD card into a USB adapter, plug it into open USB slot, and check the filesystems again:

`df -h`

There is now a new line of output at its end:

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

(The file name 6661-3738 will be different for different SD cards.)

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

`umount /dev/sda1`

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

Next you need to copy the entire system with its two partitions onto your 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.
(If your earlier df -h test showed /dev/sdc1 instead of /dev/sda1, use /dev/sdc instead of /dev/sda .)

sudo dd bs=4M if=2016-05-10-raspbian-jessie.img of=/dev/sda

This took several minutes 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        63M   21M   43M  33% /media/craig/boot
/dev/sda2       3.6G  3.2G  177M  95% /media/craig/e093a5bb-b180-4f87-9d60-467b3e79811d
```

Don't worry that the space used is less than the capacity of the SD card. This will soon be fixed.
Also, the sizes shown will be slightly different with newer versions of Raspbian.

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 keyboard
and video connected.
(The older Raspberry Pi 2 had a spring-loaded SD card, but the Raspberry Pi 3 has a friction slot
instead because people were having their cards accidentally ejected with the spring-loaded arrangement.)

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 preferences:

```
Set Localization/Set Locale to en (English), US (USA), UTF-8
Set Localization/Set Timezone to US, Pacific-New
Set Localization/Set Keyboard to United States, English (US)
    (Currently, there appears to be a bug that requires you to scroll the choices to the
     top to find the “English (US)” selection which is above “Spanish (Latin America)”.)
Set WiFi Country to US
```

(A configuration option to expand Filesystem is not needed any more. It is done automatically
at the first boot. Doing df -h should now show that the rootfs has (nearly) the full size of the SD card.)

A reboot will now be necessary for these choices to take effect.

The following note about slow mouse movement is probably no longer needed.

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=serial0,115200 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=serial0,115200 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)

wlan0     Link encap:Ethernet  HWaddr b8:27:eb:b9:8a:b3
          inet6 addr: fe80::c837:e9f0:9719:f41f/64 Scope:Link
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:15 errors:0 dropped:15 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3848 (3.7 KiB)  TX bytes:0 (0.0 B)
```

To check this Ethernet 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`

If your are doing a WiFi connection, you need to tell the Pi your WiFi ssid and psk as follows:

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"
}
```

To save nano requires you to do a <ctrl-O> (^O) , confirm the filename using <enter>, and then to quit using a <ctrl-X> (^X).

After a reboot (Menu/Shutdown/Reboot or “sudo reboot” in the command-line window), it should be able to connect to the WiFi network with the specified ssid and psk. The WiFi icon on the right side of the top line allows choices and information.

Most school and corporate connections are likely to require more items in the network clause.
(If you have chosen the US,International keyboard, this will not work because the double quotes will
be a different character, ones that look more like high double dots. You need to choose the straight US keyboard.)

Raspbian now boots straight into the graphic window mode so the following note is no longer applicable
unless you switch to a pure console mode by doing ctrl-alt-fx where x=3,4,5,or 6. ctrl-alt-f7 gets you back to
the graphic desktop mode.

```
The console modes default to assuming the keyboard is the layout for Great Britain (gb)
rather than for the United States (us).  To fix this and avoid frustration with characters
like '|', '@', and others, Edit the file /etc/default/keyboard as follows:

sudo nano /etc/default/keyboard

and change the line that says XKBLAYOUT="gb" to XKBLAYOUT="us", save your change and reboot
for the change to take effect.

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. Another reboot may be necessary if the dist-upgrade brings in
a new kernel or boot-loader.

The following note is not longer applicable. Its operation is handled by the dist-upgrade command shown above.
I am only mentioning it here to prepare you to ignore old instructions about rpi-update.

```
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.
```

The default contents of the file **/etc/network/interfaces** contained the following:

```
auto lo
iface lo inet loopback

iface eth0 inet manual

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

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

and allow graceful network connections without any changes. We do not need to edit it.
If you have both WiFi and an Ethernet cable, the Ethernet will be used unless it cannot
reach your desired destination.

**** 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
```

The following note is no longer applicable because the system boots into X-Window (graphics) mode as installed, but it is useful information if you change it to boot to console mode instead.

```
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 that might be called "**Remmina Remote Desktop**" on the program menu. On a Windows system, the choice might be called "**Real VNC**".

On my Ubuntu system, I selected “Applications/Internet/Remmina Remote Desktop Client”, chose “Connection/Add”, gave it a name “Raspberry Pi 6”, left Group blank, selected “VNC-Virtual Network Computing” protocol, did “Basic” setup giving it 10.0.0.6:1 for server, “pi” for user name, “password” for password, and left the remaining at their default settings. If you want to change the passwords, you can run a program called vncpasswd to change it.

To get the desktop shown on another Raspberry Pi, you should install xtightvncviewer as follows:

```
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.

The suggested procedure is at

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

which 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
```

At the end, add the following line to instruct vim to show line numbers when it starts up. For this, you need to be in “INSERT” mode or strange things happen. To get into INSERT mode type the letter “i” and the word “INSERT” will appear in the lower left corner. Then, add the following line at the end:

```
set number
```

To finish, you need to leave INSERT mode by pressing the <Esc> key which will turn off the word “INSERT”, and then enter ":wq" (without the quotes) which will write the file (w) and quit the vim editor (q).

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

The python imaging library PIL is now installed by default, but the following installs its documentation:

**** 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
sudo apt-get install python-matplotlib
sudo apt-get install python3-matplotlib
```

**** 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 gif images

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

It is then useful to create a directory Avogadro and put links in it to the sample molecular data:

```
mkdir /home/pi/Avogadro
cd /home/pi/Avogadro
ln -s /usr/share/avogadro/fragments/
ln -s /usr/share/avogadro/builder/
```

Avogadro can use OpenGL, but the OpenGL driver for Raspbian is still experimental and will interfere with other software (like the omxplayer) that uses the hardware graphics acceleration. It helps Avogadro run much more smoothly, however. To enable or disable it, use the program raspi-config as follows:

sudo raspi-config

Go to "**9. Advanced Options**" and "**AB. Open GL Driver**" and enable it. Then reboot.

You will then be able to run Avogadro very smoothly, but only in a reduced resolution mode of 1152x864.

Also, I had been using the 800x480 Raspberry Pi Foundation Display and the switch wiped out my "ignore_LCD=0" line or perhaps simply the new system only runs on the HDMI display. It did boot to the HDMI display and there I changed /boot/config.txt by adding the line "ignore_LCD=1".
Later, I'm not sure why, it refused to boot and I was only able to boot after disabling the 800x480 display by removing its jumpers and plugging the power directly into the pi.
Comments by others suggest that HDMI sound no longer works and the omxplayer does not work when the experimental OpenGL driver is enabled. For now, I will only enable it when I want to use Avogadro.

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

sudo apt-get install gelemental

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

sudo apt-get install evince

**** tcpdump **** A program that can monitor network traffic

sudo apt-get install tcpdump

Setting up Programming Tools and Installing Example Programs

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

```
mkdir Programming
cd Programming
```

We will end up with subdirectories named Assembly, C, C++, Python, and Java, but these will be created automatically by untaring matching tar files. While still in the /home/pi/Programming directory do the following 7 transfers:

Some notes about computer programming (1.6 MB compressed)
wget http://yosemitefoothills.com/ACM_Workshop_Notes/SomeNotesOnProgramming.tar.gz
tar xf SomeNotesOnProgramming.tar.gz
du -sb SomeNotesOnProgramming
1794833 SomeNotesOnProgramming

Assembly examples (44 MB compressed)
This will take a long time to download because of some large animations that are included.
wget http://yosemitefoothills.com/ACM_Workshop_Notes/Assembly.tar.gz
tar xzf Assembly.tar.gz
du -sb Assembly
59896202 Assembly

This will create the directory /home/pi/Programming/Assembly with various subdirectories and files. The du command reports the number of bytes in the newly created directory Assembly.

Follow a similar process for the other programming language directories, but keep in mind my yosemitefoothills.com server has a rather poor upload speed. It may be best to get them from me during the workshop.

C examples (123 kB compressed)
wget http://yosemitefoothills.com/ACM_Workshop_Notes/C.tar.gz
tar xzf C.tar.gz
du -sb C
513570 C

C++ examples (245 kB compressed)
wget http://yosemitefoothills.com/ACM_Workshop_Notes/C++.tar.gz
tar xf C++.tar.gz
du -sb C++
933991 C++

Java examples (915 kB compressed)
wget http://yosemitefoothills.com/ACM_Workshop_Notes/Java.tar.gz
tar xf Java.tar.gz
du -sb Java
1552639 Java

Python examples (82 MB compressed)
This will take a long time to download because of some large animations that are included.
wget http://yosemitefoothills.com/ACM_Workshop_Notes/Python.tar.gz
tar xf Python.tar.gz
du -sb Python
102564948 Python

Programs for video conferencing Pi-to-Pi (1.8 MB compressed)
wget http://yosemitefoothills.com/ACM_Workshop_Notes/Pi2Pi.tar.gz
tar xf Pi2Pi.tar.gz
du -sb Pi2Pi
2268631 Pi2Pi

The programs cc, c++, python, python3, java, javac, appletviewer, javadoc, etc. and important code libraries are already included in the Raspbian system.

We might as well also install some pictures, music and notes files from yosemitefoothills.com into /home/pi/:

Pictures examples (4 MB compressed, compression was largely redundant since the pictures are .jpg compressed already)
cd /home/pi
wget http://yosemitefoothills.com/ACM_Workshop_Notes/Pictures.tar.gz
tar xf Pictures.tar.gz
du -sb Pictures
4185877 Pictures

Music examples (24 MB compressed)
cd /home/pi
wget http://yosemitefoothills.com/ACM_Workshop_Notes/Music.tar.gz
tar xf Music.tar.gz
du -sb Music
25581528 Music

Workshop notes (1.2 MB compressed)
cd /home/pi
wget http://yosemitefoothills.com/ACM_Workshop_Notes/WorkshopNotes.tar.gz
tar xf WorkshopNotes.tar.gz
du -sb WorkshopNotes
1393689 WorkshopNotes

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 /boot/config.txt change to take effect.

Also, there is a desktop tool at /Menu/Preferences/Audio Device Settings/ where you need to enable the sound capability of the processor sound system. Select “Sound Card” to be “BCM2835 ALSA (Alsa mixer) (Default)” then click on “Select Controls” in the “Playback” tab, and mark the “PCM” checkbox. Close by clicking “OK”.

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. (If your are using a USB sound adapter, the choices will be a bit different.)
ls c
With a speaker connected to your HDMI output, you can then check that the sound is working by doing

```
aplay /home/pi/Programming/Java/originals/sounds/startup.wav
```

If you hear a voice, you are ready to try the Java programs.

Useful instructions for the Java example programs are at /home/pi/Programming/Java/JavaNotes-README:

```
cd ~/Programming/Java/
view JavaNotes-README
```

(view is like vi except that it prevents you from changing the file.)

**** 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/

When we untarred Assembly.tar.gz, the assembler and related files including documentation for the Atmel ATmega328P processor were placed in the directory /home/pi/Programming/Assembly so the following notes are unnecessary:

> There is an avra assembler available using apt-get install:
>
> **sudo apt-get install avra**
>
> The success of this process can be tested by doing:
>
> **avra**
>
> and seeing it produces its version and help information.
>
> 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. We want to place it in the /home/pi/Programming/Assembly directory so we first do
>
> **cd /home/pi/Programming/Assembly**
>
> The m328Pdef.inc file can be copied from github.com using
>
> **wget https://github.com/DarkSector/AVR/raw/master/asm/include/m328Pdef.inc**
>
> Later, you may want to place copies closer to your assembly code.

In order to communicate to the Gertboard via an spi connection, we need a python module named **spidev**.
We install versions of it for both the older python and newer python3 interpreters as follows:

```
sudo apt-get install python-spidev
sudo apt-get install python3-spidev
```

The success of this process can be seen by starting the python and python3 interpreters and making sure that they can import the module spidev:

```
python
Python 2.7.9 (default, Mar  8 2015, 00:52:26)
[GCC 4.9.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import spidev
>>>
(<ctrl-D> exits)
```

```
python3
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[GCC 4.9.1] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import spidev
>>>
(<ctrl-D> exits)
```

At some point apt-get install started to tell me that libftdi1 was no longer necessary and that I should run the following:

```
sudo apt-get autoremove
```

Setting Up a WiFi Access Point

These instructions are from (with slight adjustments for my ip address arrangement)

https://frillip.com/using-your-raspberry-pi-3-as-a-wifi-access-point-with-hostapd/

The next 14 lines of instructions involving brcm can be skipped. These firmware drivers now appear to be built into the Raspbian system:

```
cd /home/pi
wget https://github.com/RPi-Distro/firmware-nonfree/raw/master/brcm80211/brcm/brcmfmac43430-sdio.bin
wget https://github.com/RPi-Distro/firmware-nonfree/raw/master/brcm80211/brcm/brcmfmac43430-sdio.txt
cd /boot
sudo mkdir -p firmware/brcm/
sudo cp /home/pi/brcm* firmware/brcm/
```

As a check do

```
ls l /boot/firmware/brcm
total 376
-rwxr-xr-x 1 root root 368957 Jun  1 00:54 brcmfmac43430-sdio.bin
-rwxr-xr-x 1 root root   1108 Jun  1 00:54 brcmfmac43430-sdio.txt
```

(Upon rebooting, these will be added to /lib/firmware/brcm)

sudo apt-get install hostapd dnsmasq

Edit **/etc/default/hostapd** as superuser (sudo) to specify location of configuration file changing

`#DAEMON_CONF=""`

to

DAEMON_CONF="/etc/hostapd/hostapd.conf"

Edit **/etc/hostapd/hostapd.conf** as superuser using sudo so that it contains the following lines except that your system will use a different ssid and wpa_passphrase:

```
interface=wlan0
driver=nl80211
ssid=ACM_Workshop
hw_mode=g
channel=1
ieee80211n=1
wmm_enabled=1
ht_capab=[HT40][SHORT-GI-20][DSSS_CCK-40]
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_key_mgmt=WPA-PSK
wpa_passphrase=Lets_learn_some_stuff
rsn_pairwise=CCMP
```

We once spent a long time finding our that a passphrase for `hostapd` was too short. No error message appeared in the log. What we should have done was try to run the `hostapd` daemon by hand as follows:

sudo /usr/sbin/hostapd -B -P /run/hostapd.pid /etc/hostapd/hostapd.conf

(This line can be found by doing `ps aux | grep hostapd` on a working system if available.) In order to start the new one from the command line, any running `hostapd` must, of course, be killed by doing

sudo kill -9 <process id #>

and then waiting a few seconds to allow it to die gracefully.

Then the following message would appear if the passphrase was too short:

```
Configuration file: /etc/hostapd/hostapd.conf
Line 10: invalid WPA passphrase length 6 (expected 8..63)
WPA-PSK enabled, but PSK or passphrase is not configured.
2 errors found in configuration file '/etc/hostapd/hostapd.conf'
Failed to set up interface with /etc/hostapd/hostapd.conf
Failed to initialize interface
```

With a sufficient passphrase, the startup message says:

```
Configuration file: /etc/hostapd/hostapd.conf
Using interface wlan1 with hwaddr b8:27:eb:b9:8a:b3 and ssid "ACM_Workshop"
wlan1: interface state UNINITIALIZED->ENABLED
wlan1: AP-ENABLED
```

The minimum passphrase length is likely to be different for different choices of encryption schemes.

Edit **/etc/dnsmsasq.conf** as superuser using sudo so that it has the following lines as shown without any '#' in front of them:

```
domain-needed
bogus-priv
server=/192.168.10.6/192.168.2.3
interface=wlan0
listen-address=192.168.10.6
bind-interfaces
dhcp-range=192.168.10.20,192.168.10.31,12h
dhcp-host=80:1f:02:f6:77:54,192.168.10.4
dhcp-host=80:1f:02:a2:f2:34,192.168.10.5
dhcp-host=b8:27:eb:ca:ee:fe,192.168.10.7
```

The server line will have a forward slash, the address you choose for your access point, another forward slash, and the address of the router supplying your connection to the Internet.

The listen-address is the address selected for this access point.

The dhcp-range line is for all other computers that might connect. In your case, the server line identifies where to forward domain name requests.

The last 3 lines are because I like to give specific computers specific addresses. Those lines connect the MAC addresses of their WiFi interfaces with the IP addresses I wish them to always have.

Edit **/etc/network/interfaces** as superuser using sudo so that the original wlan0 stanza is commented out with initial '#' characters and now reads

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

We replace those lines with a wlan0 stanza that specifies the static address of the access point:

```
auto wlan0
iface wlan0 inet static
    address 192.168.10.6
    netmask 255.255.255.0
```

Reboot and ifconfig wlan0 should show

```
ifconfig wlan0
wlan0     Link encap:Ethernet  HWaddr b8:27:eb:b9:8a:b3
          inet addr:192.168.10.6  Bcast:192.168.10.255  Mask:255.255.255.0
          inet6 addr: fe80::c837:e9f0:9719:f41f/64 Scope:Link
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

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 dnsmasq enable
```

On a receiving Raspberry Pi with a receiver installed, the file /etc/wpa_supplicant/wpa_supplicant.conf file needs to be edited to have a network stanza added at its end. For this case, the network stanza looks like

```
network={
   ssid="ACM_Workshop"
   psk="Lets_learn_some_stuff"
}
```

The **iwlist** command is used to search for available wireless access points:

```
iwlist scan
```

We would like our access point to share its Internet access coming in via its Ethernet connection. This is called IP forwarding together with network address translation (NAT). They are enabled as follows:

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

This sets a flag in the kernel that immediately enables IP forwarding.

Edit **/etc/sysctl.conf** to remove the # from the start of the line (currently line 28):

```
#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 v1.4.21 on Wed Jun  1 14:39:11 2016
*filter
:INPUT ACCEPT [41:2132]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [52:4880]
-A FORWARD -i eth0 -o wlan0 -m state --state RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -i wlan0 -o eth0 -j ACCEPT
COMMIT
# Completed on Wed Jun  1 14:39:11 2016
# Generated by iptables-save v1.4.21 on Wed Jun  1 14:39:11 2016
*nat
:PREROUTING ACCEPT [0:0]
:INPUT ACCEPT [0:0]
:OUTPUT ACCEPT [1:172]
:POSTROUTING ACCEPT [0:0]
-A POSTROUTING -o eth0 -j MASQUERADE
COMMIT
# Completed on Wed Jun  1 14:39:11 2016
```

To make this iptables firewall/network translation become active at boot time, we place a file called iptables in /etc/network/if-pre-up.d:

```
sudo vi /etc/network/if-pre-up.d/iptables
```

in which we put the following two lines

```
#!/bin/sh
```

```
iptables-restore < /etc/iptables.ipv4.nat
```

And to save any changes we might make to the iptables setup upon shutdown, we place a similar file also named iptables, but with different contents into /etc/network/if-post-down.d:

```
sudo vi /etc/network/if-post-down.d/iptables
```

in which we put the following two lines

```
#!/bin/sh
iptables-save > /etc/iptables.ipv4.nat
```

Both these last two files must be made executable by doing

```
sudo chmod +x /etc/network/ip-pre-up.d/iptables
sudo chmod +x /etc/network/ip-post-down.d/iptables
```

Now, all WiFi connected computers can get an IP address from our access point. Those that have not been given specific addresses using the dhcp-host lines in /etc/dnsmasq.conf will be offered a temporary address in the range 192.168.10.20 to 192.168.10.31.

The dnsmasq daemon also works with a file /etc/hosts that helps associate names with IP addresses.
Currently, I have computers raspberrypi4, raspberrypi5, raspberrypi6 (the access point), and raspberrypi7. I can access these by name by putting their addresses and corresponding names into /etc/hosts as follows:

```
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

192.168.10.4       raspberrypi4
192.168.10.5       raspberrypi5
192.168.10.6       raspberrypi6
192.168.10.7       raspberrypi7
```

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

The masquerading provided by iptables will allow all of the computers connected to our access point to browse the Internet, but its firewall instructions will prevent incoming new connections from eth0 to penetrate into our local network.

Changing hostname and password, Enabling SPI and SSH Communication and Optionally Enabling the Camera

Using the menu, you should select Menu/Preferences/Raspberry Pi Configuration and then under the System tab change your hostname and password. (**raspberrypi6** and **ACMWorkPiB208** for this workshop.)

Also, under the Interfaces tab, SSH should already be enabled, but SPI will need to be enabled for our microprocessor programming later in the workshop.

If you have a camera, it needs to be enabled here also. The newer 8 MB cameras require 128 MB of RAM instead of 64 MB for the original 5 MB cameras. As a result, you may need to run **sudo raspi-config**, select "9 Advanced Options", then "A3 Memory Split", and enter 128 MB if you have 8 MB camera. (I am not sure if the Menu/Preferences/Raspberry Pi Configuration/Interfaces/Camera enable option checks your camera and takes care of the memory split automatically.)

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 apache2-doc
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 red line stating "It Works!". The source for that page is at /var/www/html/index.html. You can then place your 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
```

Subdirectories are made by doing

```
sudo mkdir <name of subdirectory>
```

php code is inserted into normal web pages by starting with a line

```
<?php
```

and ending with one containing

```
?>
```

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.