Raspberry Pi (RPi) Setup Documentation

Instrumentation Development Lab
Department of Physics and Astronomy
University of Hawaii at Manoa

AUTHOR: Bronson Edralin
DATE: 9/11/14

Table of Contents

1 Introduction

This documentation describes how you would set up the RPi using a spare monitor, keyboard and mouse. If you do not have one lying around, you might want to do a Headless Installation. What does Headless mean? This means that you can install the Linux Operating System (OS) directly on the SD Flash Card, without the SD Flash Card attached to the RPi. Here are some links for the Headless Installation:

Headless Installation Documentation (Alternative)

- Long Version
  - http://www.robertawood.com/blog/raspberry-pi/raspberry-pi-initial-setup-headless-no-monitor-or-keyboard-needed/raspberry-pi-initial-setup-part-6-actually-writing-the-image-to-disk.html
- Short Version
  - http://www.tristancollins.me/computing/headless-raspbian-installation/
  - http://forgetfulprogrammer.wordpress.com/2013/06/03/raspberry-pi-headless-installation/

Note: If you have a spare monitor, keyboard and mouse, you do not have to do the Headless Installation.

2 Initial Setup

2.1 Hardware

In order to set up your RPi, you must have a list of hardware and Cables:

1) Hardware
   a) USB Keyboard and USB Mouse
   b) Monitor
   c) RPi Power Supply (or phone charger)
2) Cables
   a) HDMI (for Monitor)
   b) Ethernet Cable

Figure 1: RPi with SD Flash Card and all Cables attached.

The RPi is basically a linux computer with a monitor. Initially, you will need to use a mouse, keyboard and monitor to set up your RPi. After you are done with setting it up, you won't need the mouse, keyboard and monitor anymore. You will simply access your RPi by SSHing (Secure Shell) remotely into your RPi from another computer that has its own monitor.

2.2 Choosing Operating System (OS)

Choose Debian (Wheezy) Linux OS, because it is the most popular OS. This means you can get great support online.

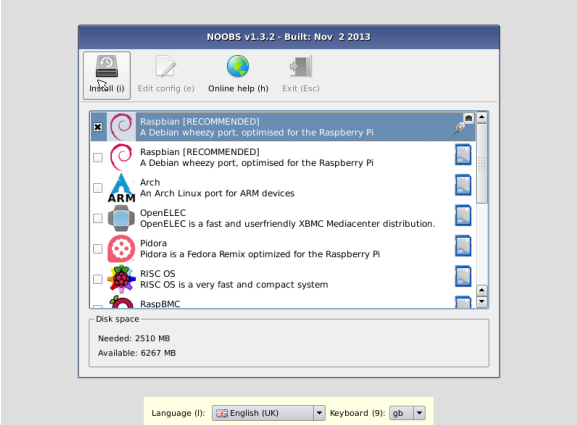

Figure 2: Choosing your Operating System (OS).

2.3 Internationalisation

2.3.1 Locales

Once installed, the RPi Software Config Tool GUI will pop up. First, you may change password to the default user "pi" from "raspberry" to "belle2" as the new password.

Then change the "Internationalisation Options"

- Choose "Change Locale"
  - From default "en_GB.UTF-8 UTF-8" (Great Britain UK)
  - To new "en_US.UTF-8 UTF-8" (USA)

If you forget to do that you may type this in your command line:

- `sudo dpkg-reconfigure locales`

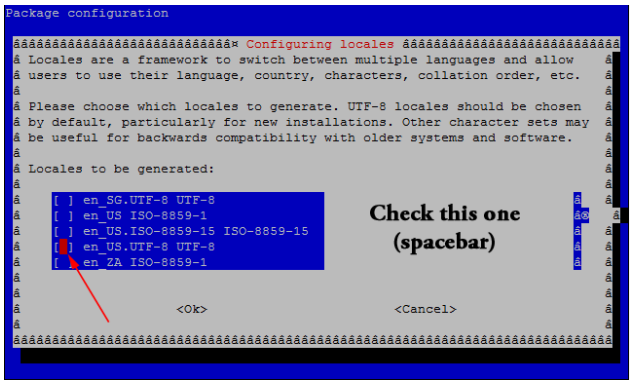

Figure 3: Need to choose your region and language.

2.3.2 Time Zone

Choose your own Time Zone. In my case, I chose Starke County (Indiana) because this was the RPi that I shipped to the University of Indiana.

If you forget to do that you may type this in your command line:

- `sudo dpkg-reconfigure tzdata`

2.4 Advanced Options

2.4.1 SSH (Secure Shell)

Enable SSH.

2.4.2 Memory Split

How much memory should GPU have? First of all, memory split divides memory between the CPU and GPU (Graphics). Most likely, we won't need too much GPU memory so we can change that to 16. If you are using your pi as a video game server or something, it probably isn't a good idea to decrease the GPU memory.

2.4.3 Hostname

Change the host name to whatever you want. I will change this one to "indiana".

If you forget to change your hostname in the initial setup, you can always do this later in the command line:

- `sudo /etc/init.d/hostname.sh start`
- `hostname`
  - >>> indiana
- `sudo find / -name hosts`
- `sudo vi /etc/hosts`
- `sudo vi /etc/hosts`
  - Then you can replace the old hostname with new host name

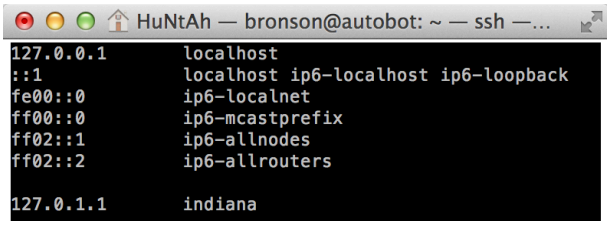

Figure 4: vi /etc/hosts

2.5 Finish and Reboot

2.5.1 Update RPi

In your command line do:

- `sudo apt-get update`
- `sudo apt-get upgrade`

3 Creating User Accounts

You will need to add user names, because the *username* will be used to SSH (or log in) into your RPi. You will also need to assign sudo access to your *username* because you need sudo access to modify the settings on your RPi. You will also need sudo access to touch the root folders.

Add User

- `sudo adduser username`
- `sudo passwd username`
- `sudo visudo`
    - Edit the sudoers file and add the user rights. In here, we are giving the users full rights. If you want, you may limit the access. There are many good material online that teaches you all about it.
    - Note: If you do "pi ALL=(ALL) NOPASSWD: ALL". By default, the user name called pi is the superuser. A superuser has the access to modify or do whatever he/she wants to the RPi. In order to give another username superuser rights, you will have to do the same as pi. Therefore, you can either do "eeuser ALL=(ALL:ALL) ALL" or "eeuser ALL=(ALL) NOPASSWD: ALL" which gives the user full access including root.

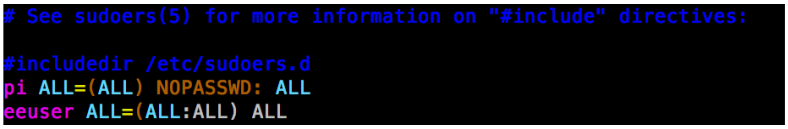

Figure 5: After typing "sudo visudo", there will be a text file that pops up for you to edit. Scroll down to the end and you will see where you can edit the sudo access.

4 Getting your Pi Online

4.1 Find out IP

By default, your pi is set to DHCP, which stands for Dynamic Host Configuration Protocol. This means that your IP may change (Note: Normally DHCP is pretty good at keeping the IP the same as long as you don't move the RPi to another port). If you move your RPi some place else, there is a potential that your IP will change. Therefore, the first thing that you must do is check to see what IP you have! On your command line you will type:

- `ifconfig`

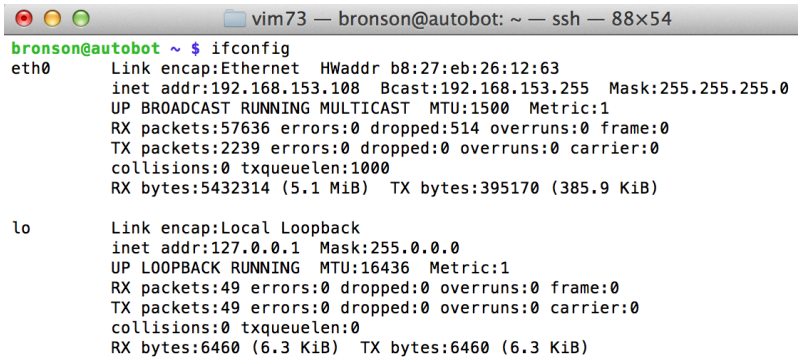

Figure 6: After typing ifconfig, the network interface configuration will be viewable.

From the results, this tells you that the configurations is:

- address 192.168.153.108 (IP you use to SSH into pi)
- netmask 255.255.255.0
- network 192.168.153.0 (network always has .0 at end)
- gateway 192.168.153.1 (gateway always has .1 at end)
- broadcast 192.168.153.255

4.2 Static IP > DHCP (Dynamic)

Note: Use this only if you want to change your IP setting from Dynamic to Static.

A Dynamic IP has risk of changing IP for your RPi so you would have to change your .ssh/config file if you want to ssh into your RPi with the new IP.

You have to edit the /etc/network/interfaces file and input the configurations and set it from dhcp to static. The /etc/network/interfaces file contains network interface configuration information for both Ubuntu and Debian Linux. This is where you configure how your system is connected to the network:

- `sudo vi /etc/network/interfaces`
    - In this file, you must make the following changes shown in the SS:

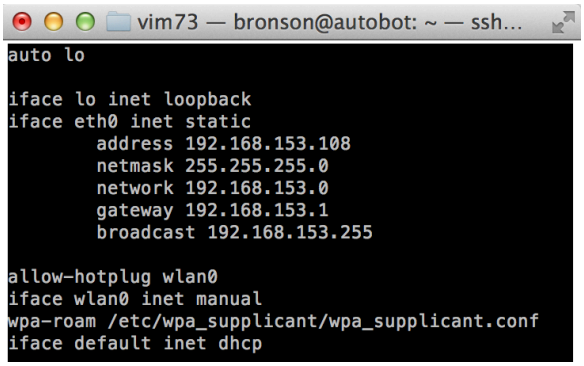

Figure 7: /etc/network/interfaces file contains network interface configuration information for both Ubuntu and Debian Linux.

Then you must do a system reboot for the changes to take effect:

- `sudo reboot`

or

- `sudo /etc/init.d/networking restart`

4.3 Connection Problems

Note: If there are connection problems, I provided some troubleshooting solutions.

On your command line, you may ping "google.com" to see if you have a working Internet connection. If it's unreachable, then you know something is wrong:

- `ping google.com`

Don't be alarmed, because problems always exist in this non-ideal world. Overall, the best solution to connection problems is to just switch to DHCP, because you get */etc/resolf.conf* for free! This file is used to configure the Domain Name System (DNS) resolver library. Using DHCP just might solve your problem, but if you still want to go with static IP, you still can.
First thing, we must do is check the *Kernal IP routing table* by typing this on the command line:

- `sudo route –n`

Your *Kernal IP routing table* should look like this:

```
bronson@autobot /home/pi/code $ sudo route -n
[sudo] password for bronson:
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.168.153.1   0.0.0.0         UG    0      0        0 eth0
192.168.153.0   0.0.0.0         255.255.255.0   U     0      0        0 eth0
```

Figure 8: Kernal IP routing table.

If you can't connect to the internet, most likely you are missing a gateway. In the *Kernal IP routing table*, it will have "*0.0.0.0*" listed under the Gateway with no actual IP listed for it. You can add a IP for the Gateway by editing the */etc/network/interfaces* file again or just doing this in command line:

- `sudo route add default gw 192.168.153.1`
    - Replace the IP for gw (Gateway) with the appropriate IP you have.

4.4 Email IP of RPi

You will have to make a python script to do this:

- `cd`
- `mkdir code`
- `cd code`
- `vi startup_mailer.py`

In *startup_mailer.py*, paste this script in there and edit the highlighted. If you are using gmail, you won't have to edit *'smtp.gmail.com',587*:

```
import subprocess
import smtplib
import socket
from email.mime.text import MIMEText
import datetime
import os
import time
import string
import urllib2
```

```
time.sleep(10)
to = 'ReplaceWithYourEmail@gmail.com'
gmail_user = 'ReplaceWithYourEmail@gmail.com'
gmail_password = 'YourPasswordToEmail'
smtpserver = smtplib.SMTP('smtp.gmail.com',587)
smtpserver.ehlo()
smtpserver.starttls()
smtpserver.ehlo
smtpserver.login(gmail_user, gmail_password)
today = datetime.date.today()
arg='ip route list'
p=subprocess.Popen(arg,shell=True,stdout=subprocess.PIPE)
data = p.communicate()
split_data = data[0].split()
ipaddr = split_data[split_data.index('src')+1]
my_ip = 'Your ip is %s' % ipaddr
msg = MIMEText(my_ip)
msg['Subject'] = 'IP For RaspberryPi on %s' % today.strftime('%b %d %Y')
msg['From'] = gmail_user
msg['To'] = to
smtpserver.sendmail(gmail_user, [to], msg.as_string())
smtpserver.quit()
```

Now you have to make the PYTHON script to automatically run every time your Pi boots up. You will have to edit the rc.local file. Point it to where the *startup_mailer.py* script lies:

- `vi /etc/rc.local`
  - Add this line, "*python /home/pi/code/startup_mailer.py*", to your *rc.local* file:

```
if [ "$_IP" ]; then
  printf "My IP address is %s\n" "$_IP"
  python /home/pi/Code/startup_mailer.py
fi
```

After you got it all set up, you should get an email of the Pi's IP whenever it boots up as shown in the picture below:

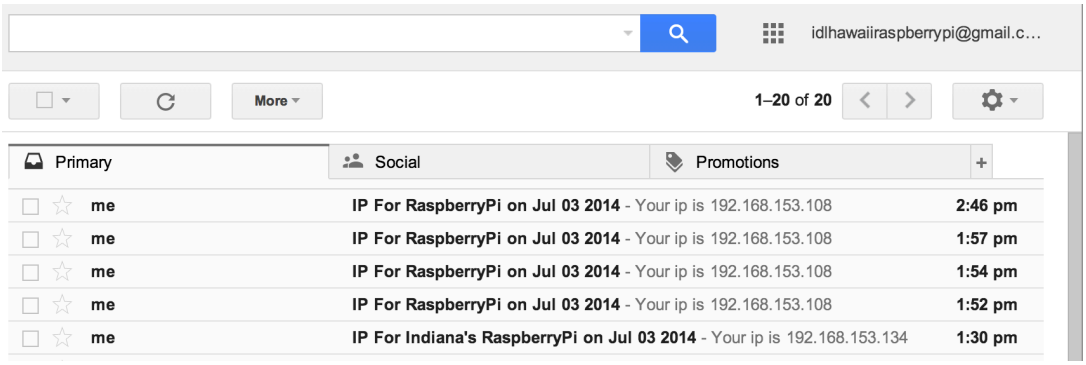

Figure 9: Sample Email Inbox that receives IP from the RPi whenever the RPi reboots.

5 VIM Colors and Features

Vim is a popular old-school *Linux Text Editor* that I personally like to use. There are other options such as nano, emacs, or gvim. This section is for someone who likes to use VIM and wants to change the colors of the text when you open up a file. Changing the colors of the text is highly useful when you start programming because it will make your code a lot more readable by distinguishing your text between "strings", "statements", etc.

5.1 Full Install Vim

Before install the full version of Vim, you must update apt-get:

- `sudo apt-get update`

5.1.1 Debian Unix OS on RPi

Debian only carries the vim lite package so you must do a full install of vim if you want to change the colors. You may install it by doing:

- `sudo apt-get install vim`

You also might want to install vim-gtk which is an enhanced vi editor compiled with GTK2 GUI. This may be used in KDE/Kubuntu-like environments. This package supports Perl, Python, Ruby and TCL scripting. You may install it by doing:

- `sudo apt-get install vim-gtk`

5.2 Change Colors

5.2.1 Debian Unix OS on RPi (Works for this particular OS)

After installing the vim packages. Then you can navigate to the folder, which contain all the different color schemes. This is where you can download and place the available colorschemes from online. Please keep in mind that *vim73* folder might have a different number such as *vim83*:

- `cd /usr/share/vim/vim73/colors`

Then you must modify the vimrc file so whenever you use vim, it will automatically set the color.

- `cd /usr/share/vim/`
- `vi vimrc`

Here's the critical part. This was a pain for me but I finally figure it out after searching through Google and trying different things. You have to edit the *vimrc* file and put syntax enable in there in order for it to work. It will not work by simply doing "*source /usr/share/vim/vim73/colors/torte.vim*". You have to put this in your *vimrc* file (Note: you may edit the color by changing the name after colorscheme. You may view the different colors in the */usr/share/vim/vim73/colors* folder):

```
syntax enable
set background=dark
colorscheme torte
```

5.3 Indentation

This is another really useful thing to do. By default, when you press the <TAB> key, it will indent by 8 spaces. This can be problematic when you start coding in PYTHON (a programming language), because PYTHON uses whitespace indentation, rather than curly braces or keywords, to delimit coding blocks. We want to decrease the amount of spaces when you press the <TAB> key.

5.3.1 Debian Unix OS on RPi (Works for this particular OS)

Then you must modify the vimrc file so whenever you use vim, it will automatically set the color.

- `cd /usr/share/vim/`
- `vi vimrc`

Then you must edit the vimrc by declaring how much spaces are associated with each indent. You may do this by putting this at the bottom of the vimrc file. In my case I wanted a tab to mean 4 spaces. This setting will be for indentation with mixed tabs and spaces:

```
set shiftwidth=4
set softtabstop=4
```

- *shiftwidth* affects what happens when you press the <TAB> key.
- *softtabstop* affects what happens when you press the <TAB> or <BS> keys. You want this to be set as the same value as *shiftwidth*.

6 SSH into your RPi

Once your RPi is set up, you can start accessing your RPi remotely with another computer. For a MAC (OSX) or Linux OS computer you will just use the default *Terminal*. If you have Windows, you will have to download a "*SSH & telnet client*" like *Putty* (http://www.putty.org/) to SSH into your RPi.

On your terminal, the commands you use will be:

- `ssh username@IP_of_your_RPi`
  - Example:
    - `ssh bronson@192.168.153.108`
    - bronson is my username, while 192.168.153.108 is the IP of the Pi that I use in the lab.

6.1 SSH CONFIG

Typing in *ssh bronson@192.168.153.108* is really bothersome. It would be a lot better to create an *alias* for the IP so we don't have to remember or type the IP of the server every time we want to access it. To create an *alias*, you have to edit the *.ssh/config* file:

- `vi .ssh/config`

Your .ssh/config file must be blank but mines look like:

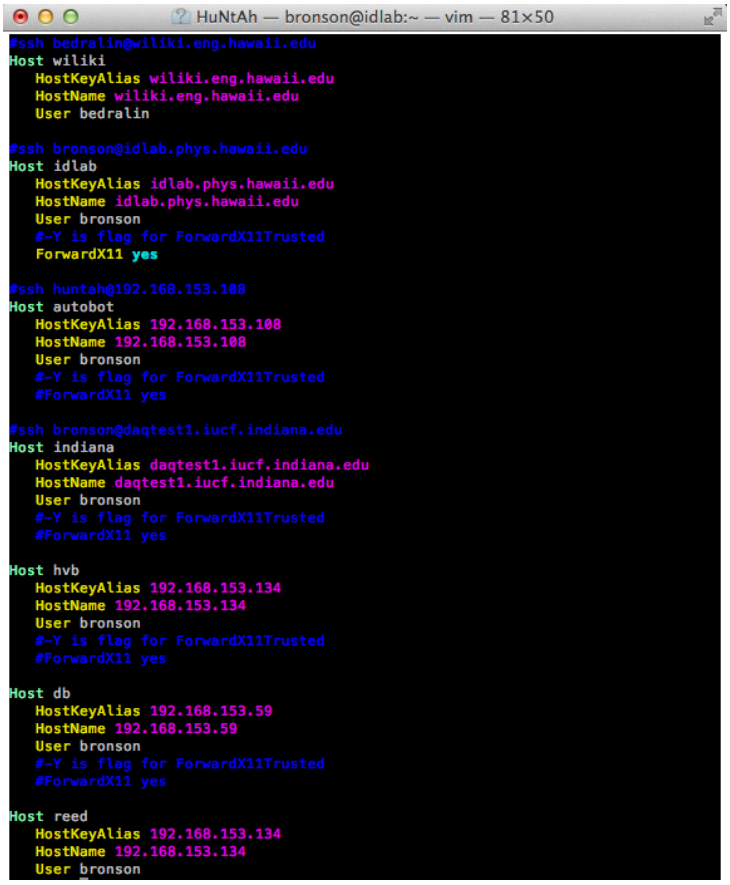

Figure 10: .ssh/config file that lets you create alias for servers IPs.

In your .ssh/config file, just paste this code and change it to your needs:

```
#ssh username@IP_Of_Your_RPi
Host alias_for_your_RPi
   HostKeyAlias IP_Of_Your_RPi
   HostName IP_Of_Your_RPi
```

```
User bronson
#-Y is flag for ForwardX11Trusted
#ForwardX11 yes
```

The "#" is used for comments. If your server uses *ForwardX11Trusted*, then you have to uncomment "ForwardX11 yes". By allowing this, when you SSH into any host that allows it, it automatically configures your X11 session and sends graphics back to your Windows workstation encrypted with SSH. You can also have X11 on your MAC (OSX). IDLab's server allows me to use Cadence Software after SSHing into the server. *ForwardX11Trusted* allowed this. Because you are setting up the RPi for the first time, it won't have those capabilities. Just comment out "ForwardX11 yes" statement.

After configuring it correctly, you should be able to SSH into your RPi remotely:

- `ssh username@alias_for_your_RPi`

Please note that the *username* is the *username* that you created when you were setting up your RPi.