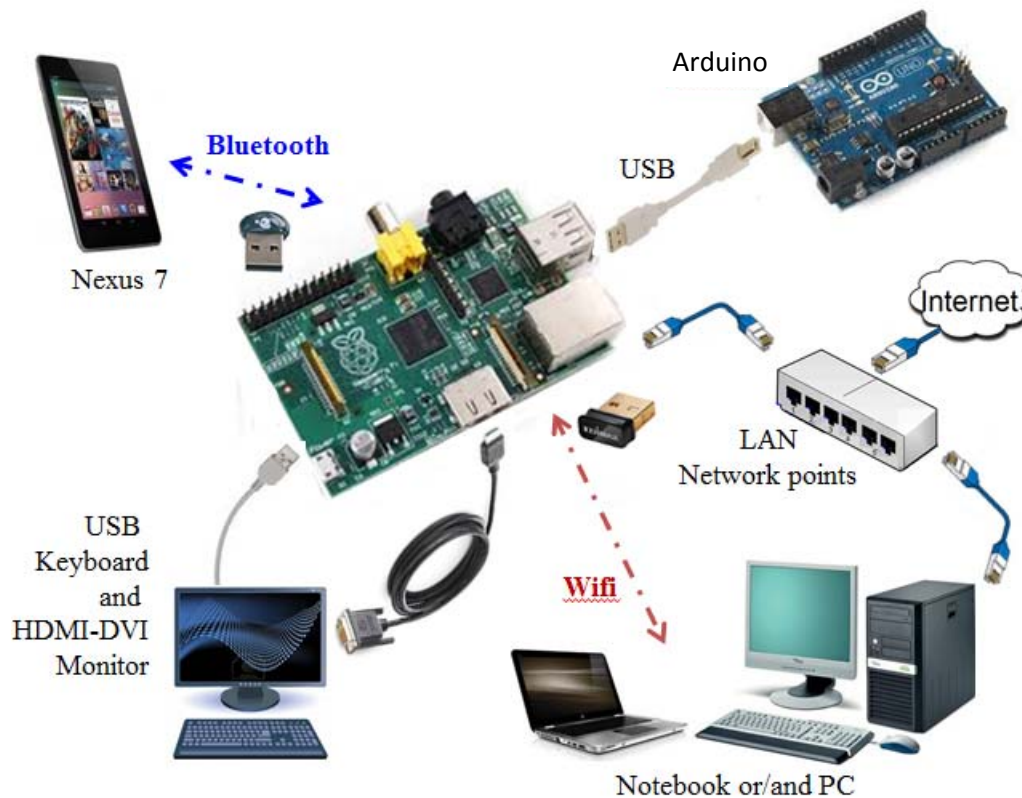


Notes on Raspberry Pi System Setup for MDP



System Overview

The Raspberry Pi (RPi) runs the Raspian¹ operating system and forms the main platform of the system, interfacing with the rest of the components as shown in figure above.

The RPi will be setup as a wireless access point such that other computing devices can interface with it through Wifi link. During system development, its wired Ethernet connection will be connected to the LAN network point in the Lab, hence also providing a gateway to the Internet for the other computing devices.

RPi's interface with the Nexus 7 (N7) tablet will be through Bluetooth connection using the rfcomm protocol. RPi interface with the Arduino will be through the USB cable using the Abstract Control Model (ACM) serial interface (ttyACM), which is similar to the USB Virtual COM but normally used when connecting to a microcontroller.

For development work, the RPi can be connected to a monitor and keyboard through its USB ports such that it can be operated as a standalone computing platform. However, due to its computing power (as well as its power supply issue – see below²), it is more convenient to access it remotely through network (either Wifi or Ethernet) using Secure Shell (SSH) remote login.

Note: Each project group will be assigned a PC in the Lab for the MDP. Students are advised to save their work in their own USB drive. The PC environment in the lab are all virtualized and all work saved in the PC will be lost after a system reboot or when you logout from the PC.

To save the files you create on RPi, you can install the Samba Server, which allow you to transfer the files over the network and store them on your own PC or thumbdrive.

Notes on Raspberry Pi System Setup for MDP**¹ Raspbian**

The Raspbian operating system used in RPi is based on the Debian Linux. Although it comes with a GUI, it is not very convenient for development work. Hence you will instead frequently **use the Linux text command based interface (remotely through SSH)**. As such, you will need to be familiar with some of the commonly use Linux commands (e.g. sudo, apt-get install, chmod etc.) as well as one of the command line text editor (e.g. nano or vi) when using the RPi.

² Power Supply issue

The RPi is setup to power all other devices through its USB port, which is in turn itself powered through its micro USB connector (S1). However, RPi contains an onboard fuse (F3, beneath the S1 on the opposite side of the PCB). This is a polyfuse whose resistance will increase as it gets hot, which is used to protect the RPi by limiting the total current allowable (about 750mA maximum) for the RPi. Beyond this current level, the F3's resistance will increase, and cause the supply voltage to drop below the operating voltage of the devices connected to the RPi. This will sometime cause seemingly 'strange' intermittence problems, whereby the device may stop operating although it is OK initially. Examples of these would be the Wifi and Bluetooth dongles. They **may stop operating after they are wireless connected**, as they start to draw more operating current that may pull the supply voltage to below their operating level. **(You can use a voltmeter to measure the supply voltage between the TP1 and TP2 testpoints on the RPi board. It is supposed to be about 5V, and should not be below 4.6V).**

As such, during the initially setup stage, you should **set up and test each interface separately** (i.e. with only one device plugged into the RPi USB port at one time). You can also **power the RPi using the N7 adaptor instead of using the PC/notebook which may have limited current in its USB port**. Hence it is also better to not use a local keyboard and monitor connected to the RPi (after the wireless SSH is setup properly^{3,5}), as they will also draw more operating current through the RPi.

³ Local monitor

The RPi will probably need to be connected to the Internet most of the time during the development work. However, due to the use of DHCP in the SCE Lab's LAN, we need to know the IP in order to use the remote SSH. Hence it is likely that you will need to connect the local monitor in order to observe the IP⁴ allocated during the bootup.

⁴ IP

If the IP address is not shown in the bootup message, you could enable it through the **/etc/rc.local** script file (which should already be enabled by default). However, sometime the DHCP server is too slow to issue the IP address, before the RPi completes the bootup, and hence will not be able to show the IP address allocated. For such case, you will need the local keyboard and monitor to login and execute the **ifconfig** command.

⁵ Wireless Access Point and Internet gateway

The most convenient setup is to not connect the local monitor (and keyboard), and be able to SSH RPi through the network. For this, we need a fixed IP address to access RPi. This can be done by setting up the RPi as a Wifi access point which uses a fixed static IP address. We can then use IP forwarding to send the IP packets to its wired Ethernet, which is connected to the LAN (and uses dynamic IP address allocated by NTU network) and form a gateway to the Internet.

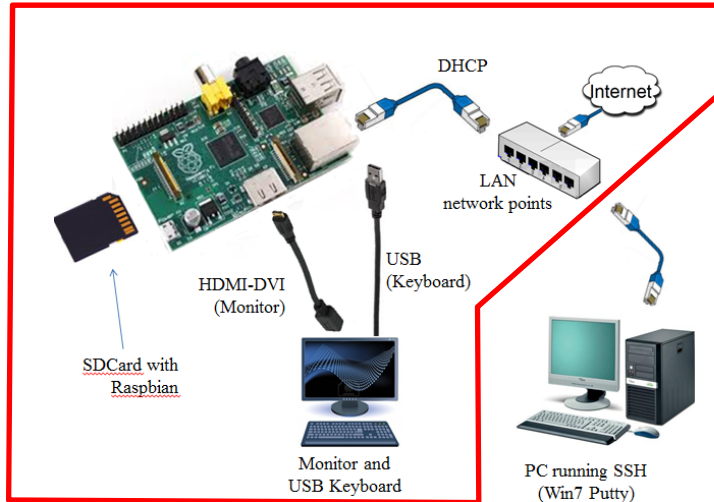
The main **content** of this document are summarized as follows:

- A. **Setting up the RPi with Raspbian**
- B. **Remote access RPi using SSH**
- C. **Setting up the RPi as a Wifi access point, and a gateway to the Internet**, which consists of 4 steps
 1. Install and configure the hostapd package to enable the Wifi dongle
 2. Configure the IP address to be used by the Wifi dongle
 3. Install and configure a DHCP server on RPi to form the Wifi access point.
 4. Configure the IP forwarding and NAT for gateway to Internet through the Ethernet connection.
- D. **Setting up the Bluetooth on RPi**
- E. **Information on the USB interface between RPi and Arduino board**
- F. **Other useful packages to have on RPi**
 1. Apache server to provide web-based GUI
 2. Samba server to transfer file over network

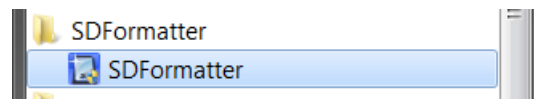
Notes on Raspberry Pi System Setup for MDP

A. Setting up of the RPi with Raspbian

The SD Card for the RPi will be installed with the Raspbian operating system, through the NOOBS (New out-of-the box software, which is a way to simplify the setting up a Raspberry Pi for the first time.) At this point, the RPi should be connected to the monitor and keyboard, as well as the network point in the Lab for internet access.

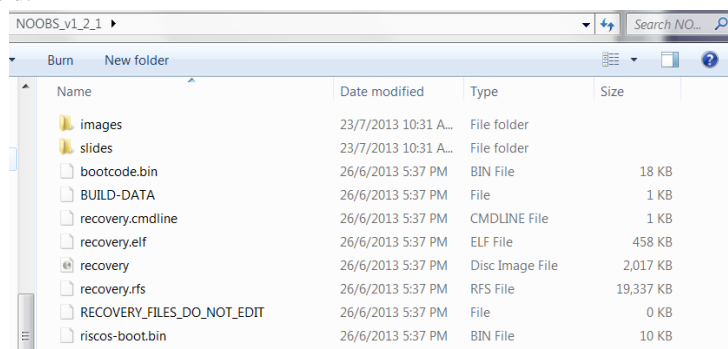


- (i) To begin, the SD card will need to be formatted first, using the Windows PC in the Lab. This is to be done using the SDFormatter program, which should already be installed on the PC in the Lab. If not download the SDFormatter program (google for it) and install it on the Windows PC.

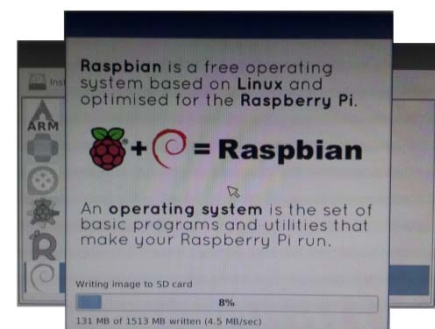


Plug the SD card into the PC. Run the SDFormatter. Select the SD card and format it.

- (ii) The NOOBS program should also be available on the PC (inside the MDP subdirectory). Copy it to your group subdirectory, unzip it and copy all the extracted files (and its subdirectory) to the formatted SD Card.



- (iii) Plug the card into RPi, apply power to the RPi through the micro USB cable. The RPi should boot up and displays messages on the local monitor, which will request the user to choose the operating system to be installed. For MDP, choose Raspbian. The RPi will then start the installation process.



Notes on Raspberry Pi System Setup for MDP

- (iv) After the installation is complete, the RPi will reboot. On the first boot, the user will be prompted with a configuration tool called Raspi-Config, for the initial configuration of the board. Select option 4 (timezone) to update, and then option 8 (hostname and SSH server, update) to enable the SSH server. It is recommended that you DON'T select the Desktop mode as it is more convenient to use the text console for the various setup steps.

Note that the default hostname, login, and password are as follows:

Hostname: raspberrypi Username: **pi** Password: **raspberry**

(You probably would want to change the **Hostname** to something that match your group, like **MDPGrpxx**)

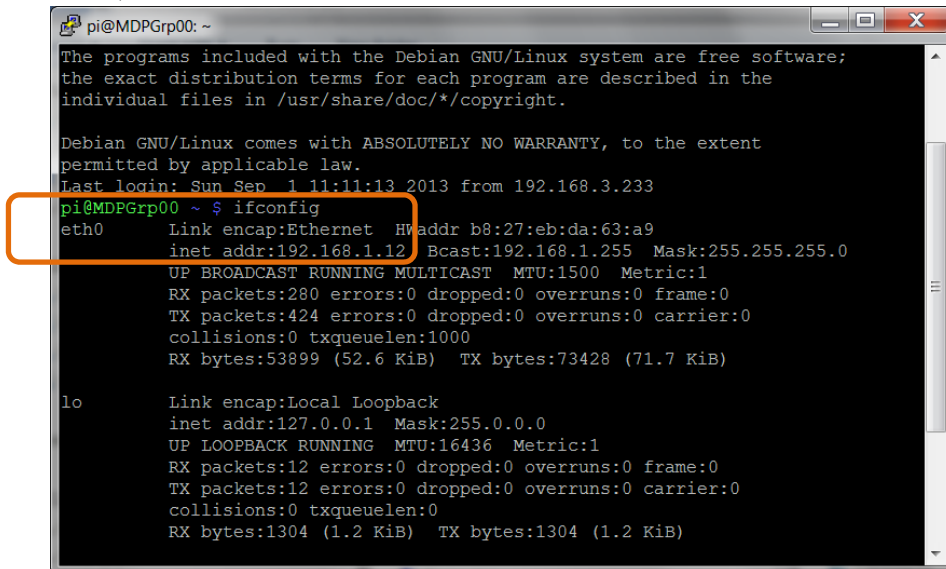
- (v) Once the configuration is done, the RPi will reboot, and should be in the Command Line terminal mode (also known as console) - unless you have select Desktop mode during the configuration in step iv. Once reboot, you have a fully functional computer (i.e. RPi board with ARM CPU) running the Linux OS (Raspbian, which is based on the Linux's Debian). As such, students will need to be familiar with some basic Linux commands in order to operate the RPi system, which will be introduced as we move along in this guide.

(Note: Raspi-Config can be subsequently executed from console using the Linux command: **sudo raspi-config** when needed. After you finish your changes to the raspi-config, you should reboot your RPi using the following Linux command: **sudo shutdown -r 0**)

Notes on Raspberry Pi System Setup for MDP

B. Remote SSH client

- (i) The first thing to check once the RPi is operating is to make sure that it has been allocated an IP address through the wired Ethernet connection to the LAN in the Lab. In the command line console, types the Linux command: **ifconfig**. Look for the lines associated with **eth0**, and check that an IP address is assigned to the RPi. (This is dynamically assigned by the NTU DHCP server through the LAN in the Lab. As this is a dynamically assigned IP, you would need to recheck this every time you reboot the RPi, as it is probably changed every time you reboot RPi – see earlier note⁴)



```
pi@MDPGrp00: ~
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun Sep  1 11:11:13 2013 from 192.168.3.233
pi@MDPGrp00 ~$ ifconfig
eth0      Link encap:Ethernet  HWaddr b8:27:eb:da:63:a9
          inet addr:192.168.1.12  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:280 errors:0 dropped:0 overruns:0 frame:0
          TX packets:424 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:53899 (52.6 KiB)  TX bytes:73428 (71.7 KiB)

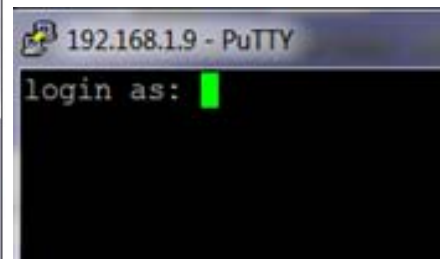
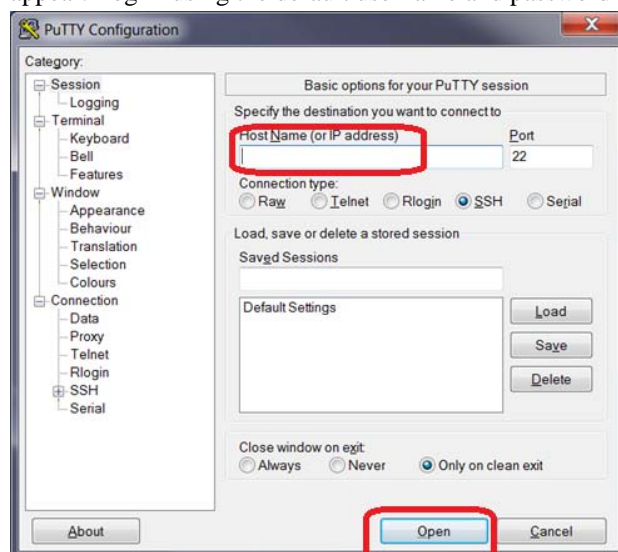
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:12 errors:0 dropped:0 overruns:0 frame:0
          TX packets:12 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1304 (1.2 KiB)  TX bytes:1304 (1.2 KiB)
```

You can also check whether your RPi has connection to Internet by using the command ‘ping’ as follows:

ping 8.8.8.8

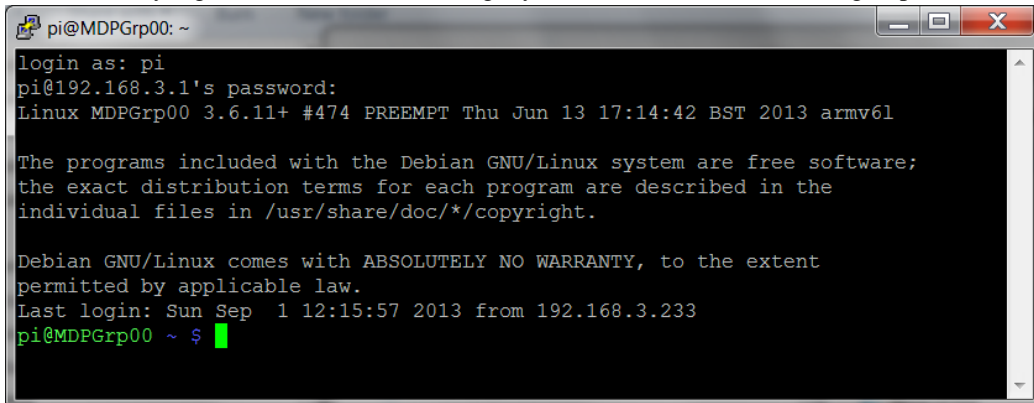
(Google to check who holds this auspicious IP address, which is a DNS computer on the internet)

- (ii) Once the eth0 IP address is known (value will depend on the LAN’s DHCP setup), you can then use another computer on the network to remotely access the RPi, using a SSH Client program. For MDP, the program [Putty](#) should already be installed on the Windows PC in the Lab. Execute the Putty program and configure it with the RPi IP address. Click Open. A console screen will appear. Log in using the default username and password (i.e. **pi** and **raspberry**).



Notes on Raspberry Pi System Setup for MDP

- (iii) You should now have remote network access to the RPi through the LAN (which means the local monitor and keyboard is not really necessary anymore.) Note that multiple SSH sessions can also be simultaneously log into the RPi if wanted (e.g. by the different members in the group)



```
pi@MDPGrp00: ~
login as: pi
pi@192.168.3.1's password:
Linux MDPGrp00 3.6.11+ #474 PREEMPT Thu Jun 13 17:14:42 BST 2013 armv6l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

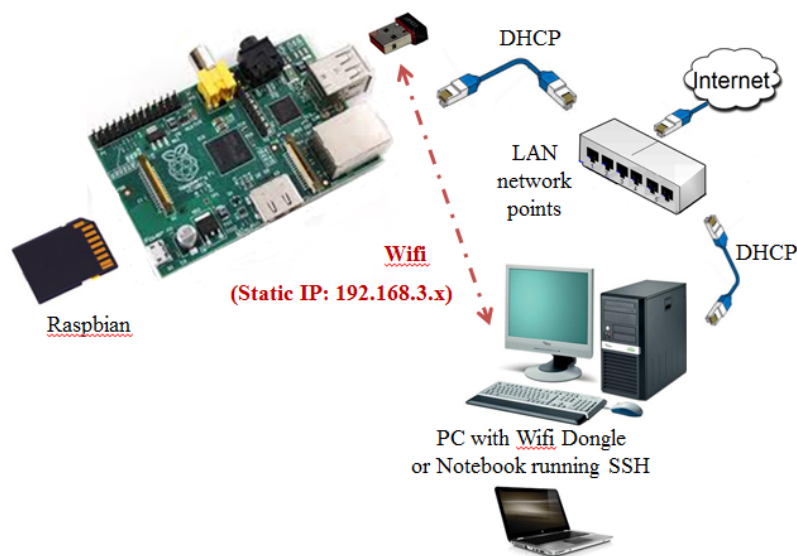
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun Sep  1 12:15:57 2013 from 192.168.3.233
pi@MDPGrp00 ~ $
```

Note: While the local keyboard/monitor is now not essential once remote SSH is established, you will still need to have the local monitor connected during the following setups such that you can observe the bootup messages to check the dynamically assigned IP^{3,4} address, and find potential causes of problems encountered. The keyboard and monitor can be disconnected once the Wifi access point and internet gateway is successfully setup⁵ eventually, as you will achieve through this guide.

C. Setting up of the RPi as a Wireless Access Point with gateway to the Internet

In the MDP proposed system, the RPi is to remotely accessible through a Wifi wireless link (as well as a Bluetooth wireless link). The section describes how the RPi can be setup as a wireless access point (i.e. a Wifi hotspot). To do so, a number of programs need to be further installed on RPi, follows by the proper network configuration.

For this section, a USB based Wifi dongle (Edimax) needs to be plugged into the RPi USB port (take note of the potential power supply issue² described on page 2). If you are allocated a desktop PC (instead of the notebook) in the Lab, which would not have built-in Wifi, another Edimax Wifi dongle should be plugged into the PC for this exercise. (The Win7 PC should automatically detect and setup the necessary driver for the Edimax Wifi dongle.) Alternatively, students can use their own notebooks (which should come with Wifi) for the following steps.



C.1 – Setting up hostapt

The main program needed to be first installed to setup the RPi as a wireless access point is: **hostapt**. This is a user space program (as opposed to kernel space program in Linux) that will run in the background (called **daemon**) and provide the wireless **host** access **p**oint service (google to find out more detail about this program).

- (i) This program can be easily downloaded and installed automatically (with the RPi connected to the internet through eth0) by simply issuing the following Linux command in the SSH client console:

sudo apt-get install hostapt

```
pi@RPiGrp0: ~
pi@RPiGrp0 ~ $ sudo apt-get install hostapt
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  hostapt
0 upgraded, 1 newly installed, 0 to remove and 12 not upgraded.
Need to get 419 kB of archives.
After this operation, 908 kB of additional disk space will be used.
Get:1 http://mirrordirector.raspbian.org/raspbian/ wheezy/main hostapt armhf 1:1.0-3 [419 kB]
Fetched 419 kB in 1s (297 kB/s)
Selecting previously unselected package hostapt.
(Reading database ... 62275 files and directories currently installed.)
Unpacking hostapt (from .../hostapt_1%3a1.0-3_armhf.deb) ...
Processing triggers for man-db ...
Setting up hostapt (1:1.0-3) ...
pi@RPiGrp0 ~ $
```

Notes on Raspberry Pi System Setup for MDP

- (ii) Once the hostapd is installed, a **hostapd** binary file would be created in a `/etc/sbin/` subdirectory as follows: **`/usr/sbin/hostapd`** (together with other necessary files, such as those found in the subdirectory **`/etc/hostapd/`**)

However, the installed copy of hostapd is **not suitable** (as at **Aug-2013**) **for operating the Edimax Wifi dongle as an access point** - we hence need to update this hostapd with a proper version. The required hostapd file needed can be download from internet, such as from the one shown below (a copy is also located in the MDP folder in NTULearn).

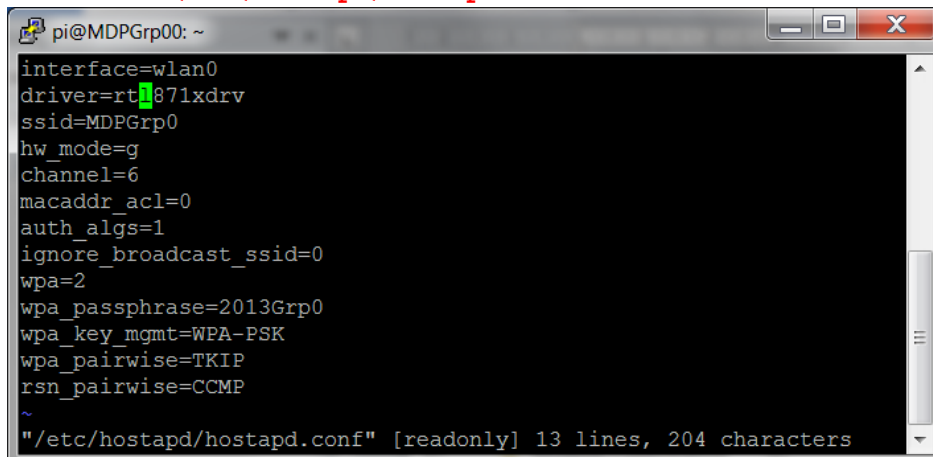
The following Linux commands can be used to **update and replace the default hostapd**, and make it an executable file. (Here, we move the original hostapd to another directory as backup, and we then symbolic link the required hostapd binary file, instead of renaming it directly)

```
wget http://www.daveconroy.com/wp3/wp-content/uploads/2013/07/hostapd.zip
unzip hostapd.zip
sudo mv /usr/sbin/hostapd /usr/sbin/hostapd.bak
sudo mv hostapd /usr/sbin/hostapd.edimax
sudo chmod +x /usr/sbin/hostapd.edimax
sudo ln -sf /usr/sbin/hostapd.edimax /usr/sbin/hostapd
```

```
pi@MDPGrp00 ~ $ sudo mv /usr/sbin/hostapd /usr/sbin/hostapd.org
pi@MDPGrp00 ~ $ sudo mv hostapd /usr/sbin/hostapd.edimax
pi@MDPGrp00 ~ $ ls -al /usr/sbin/hostapd.edimax
-rw-r--r-- 1 pi pi 1678724 Sep 27 2012 /usr/sbin/hostapd.edimax
pi@MDPGrp00 ~ $ sudo chmod +x /usr/sbin/hostapd.edimax
pi@MDPGrp00 ~ $ ls -al /usr/sbin/hostapd.edimax
-rwxr-xr-x 1 pi pi 1678724 Sep 27 2012 /usr/sbin/hostapd.edimax
pi@MDPGrp00 ~ $ sudo ln -sf /usr/sbin/hostapd.edimax /usr/sbin/hostapd
pi@MDPGrp00 ~ $ ls -al /usr/sbin/hostapd
lrwxrwxrwx 1 root root 24 Aug 31 20:06 /usr/sbin/hostapd -> /usr/sbin/hostapd.edimax
pi@MDPGrp00 ~ $
```

- (iii) Next create the following configuration file to configure hostapd. (You can use the Linux command line text editor program, **nano** or **vi** to edit it, using the command as shown below. Google 'Linux nano' or 'vi' for detail of the programs usage)

```
sudo nano /etc/hostapd/hostapd.conf
```



```
pi@MDPGrp00: ~
interface=wlan0
driver=rtl871xdrv
ssid=MDPGrp0
hw_mode=g
channel=6
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_passphrase=2013Grp0
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
~
"/etc/hostapd/hostapd.conf" [readonly] 13 lines, 204 characters
```

The configuration includes the driver (should be **RTL871XDRV**, letters all in lower case), the ssid of the Wifi hotspot and its password. It is suggested that each group use its group number within the ssid such that the Wifi signal can be clearly identified later.

Examples:

Group 1 should set the **ssid=MDPGrp1**

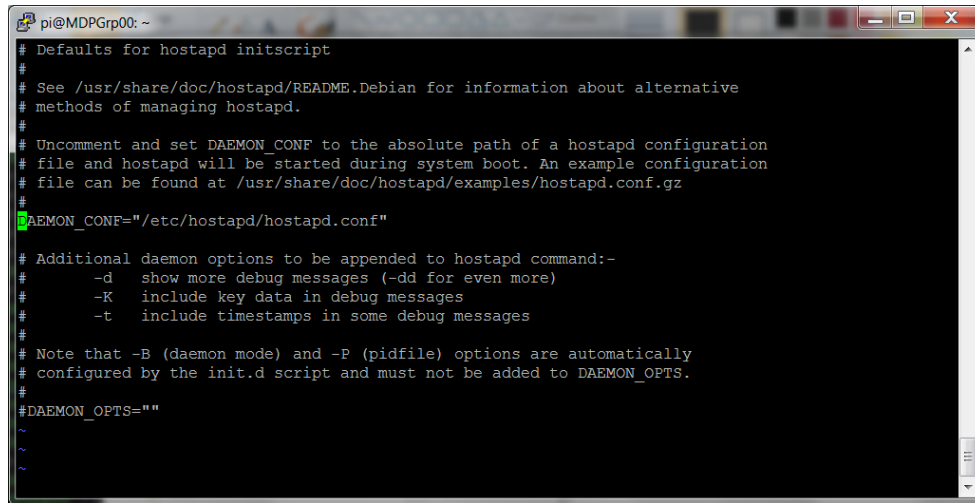
Group 3 should set the **ssid=MDPGrp3**

Notes on Raspberry Pi System Setup for MDP

- (iv) Next edit the file `/etc/default/hostapd` and change the line to point to the configuration file just created as follows:

Change `#DAEMON_CONF=""`

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



```
pi@MDPGrp00: ~
# Defaults for hostapd initscript
#
# See /usr/share/doc/hostapd/README.Debian for information about alternative
# methods of managing hostapd.
#
# Uncomment and set DAEMON_CONF to the absolute path of a hostapd configuration
# file and hostapd will be started during system boot. An example configuration
# file can be found at /usr/share/doc/hostapd/examples/hostapd.conf.gz
#
DAEMON_CONF="/etc/hostapd/hostapd.conf"
#
# Additional daemon options to be appended to hostapd command:-
#
#   -d show more debug messages (-dd for even more)
#   -K include key data in debug messages
#   -t include timestamps in some debug messages
#
# Note that -B (daemon mode) and -P (pidfile) options are automatically
# configured by the init.d script and must not be added to DAEMON_OPTS.
#
#DAEMON_OPTS=""
~
~
~
```

- (v) At this stage, you can test the access point setup by starting the hostapd using the following command:

`sudo service hostapd start`

The Wifi dongle should start flashing with blue color, showing that the hostapd is configured properly.



Before we continue, stop the access point program first.

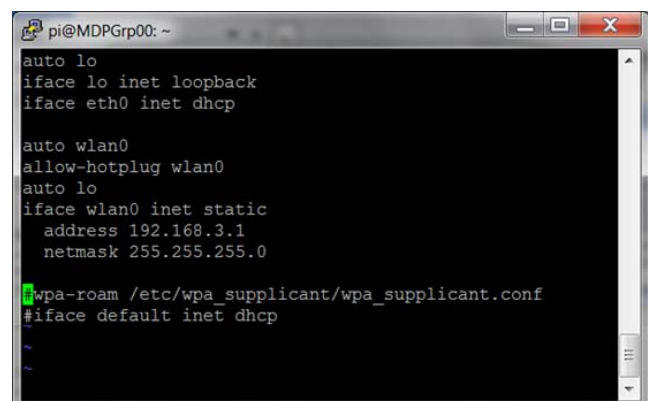
`sudo service hostapd stop`

C.2 – Setting up the Wireless interface wlan0

Before we install the DHCP, we will first assign a static IP address to the Wifi interface, which is indicated as wlan0 in Linux. (The wired Ethernet interface is referred to as eth0, as seen earlier)

- (i) Edit the network configuration for the Wifi (Wifi will be denoted as wlan0 in Linux) for wlan0 in `/etc/network/interfaces` similar to as follows (with other not used lines marked off by using the comment character #, or simply removed):

```
auto lo
iface lo inet loopback
iface eth0 inet dhcp
auto wlan0
allow-hotplug wlan0
iface wlan0 inet static
address 192.168.3.1
netmask 255.255.255.0
```



```
pi@MDPGrp00: ~
auto lo
iface lo inet loopback
iface eth0 inet dhcp

auto wlan0
allow-hotplug wlan0
auto lo
iface wlan0 inet static
address 192.168.3.1
netmask 255.255.255.0

wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf
#iface default inet dhcp
~
~
~
```

The first three lines are for the loopback and wired Ethernet eth0, where the IP will be assigned through NTU's DHCP server. The important setting is the IP for the Wifi, which must be unique among all the groups. The suggested value to be used is `192.168.group_number.group_number`, or similar

i.e. Group 1 uses 192.168.1.1

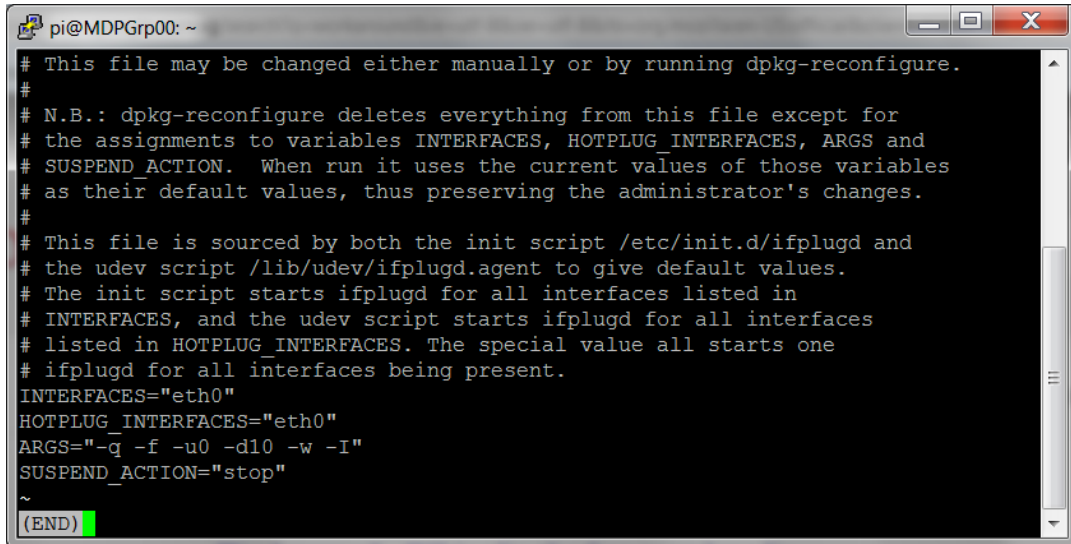
Group 2 uses 192.168.2.2 or 192.168.2.1

Group 40 uses 192.168.40.40 or 192.168.40.1

Notes on Raspberry Pi System Setup for MDP

IP addresses 192.168.x.x is reserved for 'private' network, which should not be exposed to the Internet directly (Tunnelling using IP forwarding is used instead to access Internet from private network)

- (ii) There seems to be a bug when the Wifi is set to use static IP address – when it is sometimes not set upon bootup. One workaround is to change the entries in the file `/etc/default/ifplugd` as to shown below:

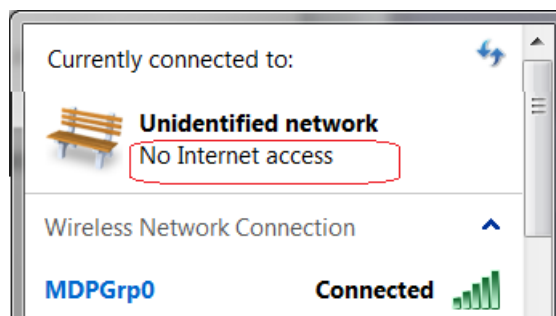


```
# This file may be changed either manually or by running dpkg-reconfigure.
#
# N.B.: dpkg-reconfigure deletes everything from this file except for
# the assignments to variables INTERFACES, HOTPLUG_INTERFACES, ARGS and
# SUSPEND_ACTION. When run it uses the current values of those variables
# as their default values, thus preserving the administrator's changes.
#
# This file is sourced by both the init script /etc/init.d/ifplugd and
# the udev script /lib/udev/ifplugd.agent to give default values.
# The init script starts ifplugd for all interfaces listed in
# INTERFACES, and the udev script starts ifplugd for all interfaces
# listed in HOTPLUG_INTERFACES. The special value all starts one
# ifplugd for all interfaces being present.
INTERFACES="eth0"
HOTPLUG_INTERFACES="eth0"
ARGS="-q -f -u0 -d10 -w -I"
SUSPEND_ACTION="stop"
~
(END)
```

- (iii) Restart the RPi (using the command `sudo shutdown -r 0`), and the network interfaces should be running and configured with IP address indicated in the interfaces file. (Check this using the command `ifconfig` after login).

The hostapd should also start running with the Wifi dongle blinking in blue color. (This should be enabled by default after the hostpad is installed. Otherwise run the hostapd service command as before)

Use a Wifi-enable PC/notebook to scan for the Wifi signal. The RPi Wifi signal probably would be detected at this stage and can be connected by the PC/notebook, but without any IP addressed being issued after connection.

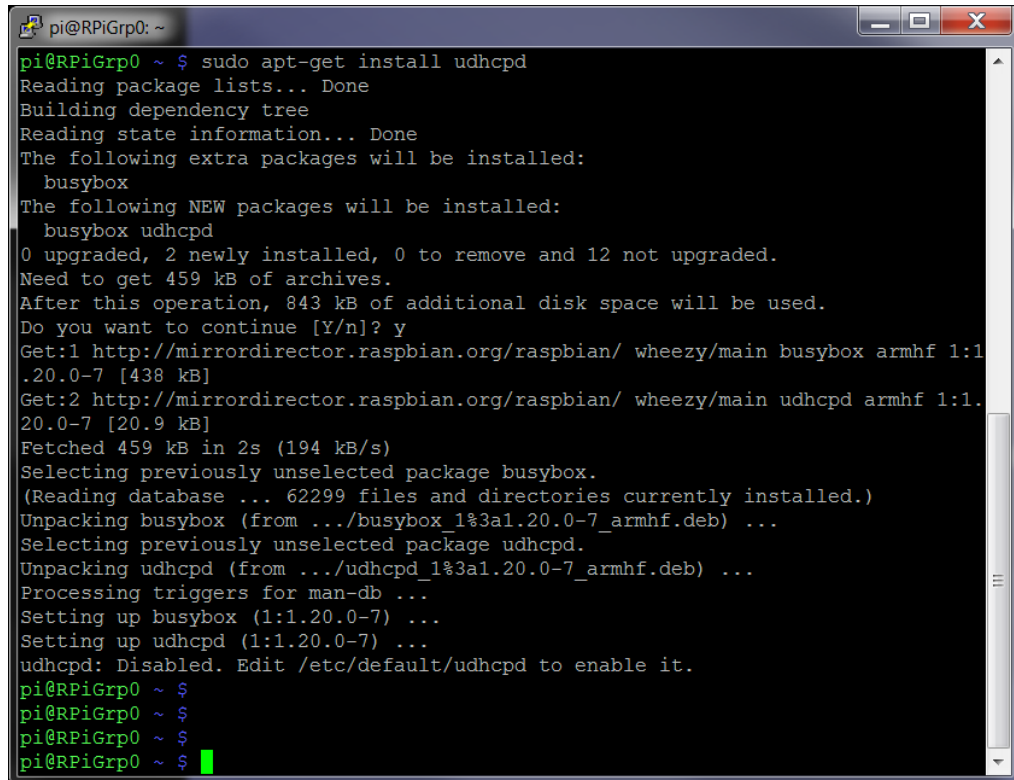


To do so, we need to setup a DHCP server on the RPi board, which we will do in the next step.

Notes on Raspberry Pi System Setup for MDP
C.3 – Installing and setting up a DHCP server for the Wifi access point (udhcpd)

The section is to setup a DHCP server on the RPi. The program that we will use here is the udhcpd (but you can use other appropriate one as you prefer – there are available through many references on RPi sites on internet)

- (i) In the SSH client console, issue the following Linux command: **sudo apt-get install udhcpd**



```
pi@RPiGrp0: ~ $ sudo apt-get install udhcpd
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  busybox
The following NEW packages will be installed:
  busybox udhcpd
0 upgraded, 2 newly installed, 0 to remove and 12 not upgraded.
Need to get 459 kB of archives.
After this operation, 843 kB of additional disk space will be used.
Do you want to continue [Y/n]? y
Get:1 http://mirrordirector.raspbian.org/raspbian/ wheezy/main busybox armhf 1:1.20.0-7 [438 kB]
Get:2 http://mirrordirector.raspbian.org/raspbian/ wheezy/main udhcpd armhf 1:1.20.0-7 [20.9 kB]
Fetched 459 kB in 2s (194 kB/s)
Selecting previously unselected package busybox.
(Reading database ... 62299 files and directories currently installed.)
Unpacking busybox (from .../busybox_1%3a1.20.0-7_armhf.deb) ...
Selecting previously unselected package udhcpd.
Unpacking udhcpd (from .../udhcpd_1%3a1.20.0-7_armhf.deb) ...
Processing triggers for man-db ...
Setting up busybox (1:1.20.0-7) ...
Setting up udhcpd (1:1.20.0-7) ...
udhcpd: Disabled. Edit /etc/default/udhcpd to enable it.
pi@RPiGrp0 ~ $
pi@RPiGrp0 ~ $
pi@RPiGrp0 ~ $
pi@RPiGrp0 ~ $
```

- (ii) A new file will also be created: **/etc/udhcpd.conf**. The appropriate lines in the file are to be edited to as shown below (**sudo nano /etc/udhcpd.conf**.) (Note that the IP addresses setting used here correspond to the 192.138.3.1 used in the Wifi's IP example earlier. Each group should hence set the IP addresses accordingly in this file)

```
# The following two lines indicate the range of IPs that the
# Wifi hotspot will give to client devices.
start 192.168.3.2
end 192.168.3.20

interface wlan0 # The device interface that udhcp listens on - Wifi.

remaining yes
opt dns 8.8.8.8 4.2.2.2 # The DNS servers client devices will use.
opt subnet 255.255.255.0
opt router 192.168.3.1 # RPi's Wifi IP address setup earlier.
opt lease 864000 # 10 day DHCP lease time in seconds
```

Notes on Raspberry Pi System Setup for MDP

```
pi@MDPGrp00: ~
# Sample udhcpd configuration file (/etc/udhcpd.conf)

# The start and end of the IP lease block

start      192.168.3.2      #default: 192.168.3.2
end        192.168.3.254    #default: 192.168.3.254

# The interface that udhcpd will use

interface   wlan0          #default: wlan0

# The maximum number of leases (includes addresses reserved
# by OFFER's, DECLINE's, and ARP conflicts

#max_leases 254            #default: 254

# If remaining is true (default), udhcpd will store the time
# remaining for each lease in the udhcpd leases file. This is
# for embedded systems that cannot keep time between reboots.
# If you set remaining to no, the absolute time that the lease
```

```
pi@MDPGrp00: ~
#boot_file    /var/nfs_root      #default: (none)

# The remainder of options are DHCP options and can be specified with the
# keyword 'opt' or 'option'. If an option can take multiple items, such
# as the dns option, they can be listed on the same line, or multiple
# lines. The only option with a default is 'lease'.

#Examples
opt   dns      8.8.8.8 4.2.2.2
option subnet 255.255.255.0
opt   router   192.168.1.1
opt   wins     192.168.10.10
option dns     129.219.13.81 # appened to above DNS servers for a total of 3
option domain  local
option lease   864000        # 10 days of seconds

# Currently supported options, for more info, see options.c
#opt subnet
#opt timezone
#opt router
#opt timesrv
#opt namesrv
```

Another file needs to be also edit: **/etc/default/udhcpd**. Comment off (i.e. disable) the following line in the file by adding a '#' in front of the line as follows.

Change **DHCPD_ENABLED="no"**
to **#DHCPD_ENABLED="no"**

```
pi@MDPGrp00: ~
# Comment the following line to enable
#DHCPD_ENABLED="no"

# Options to pass to busybox' udhcpd.
#
# -S    Log to syslog
# -f    run in foreground

DHCPD_OPTS="-S"

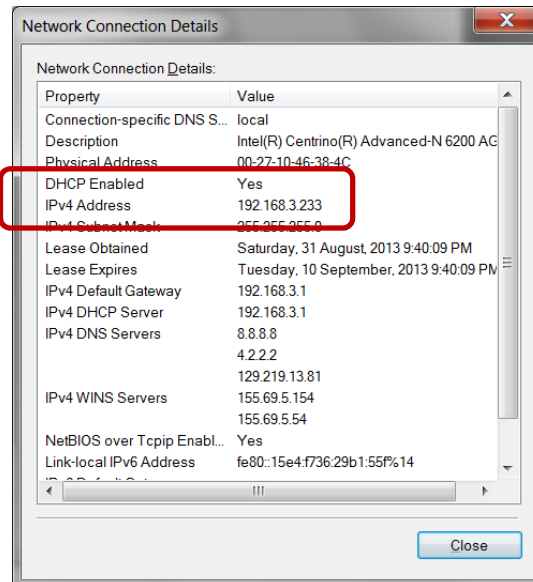
~/
"/etc/default/udhcpd" [readonly] 9 lines, 165 characters
```

Notes on Raspberry Pi System Setup for MDP

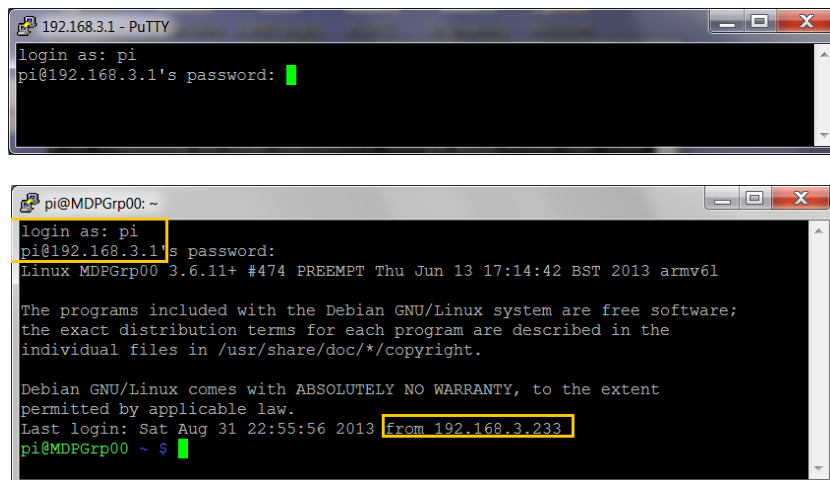
- (iii) Test the DHCP service using the following command:

```
sudo service udhcpd start
```

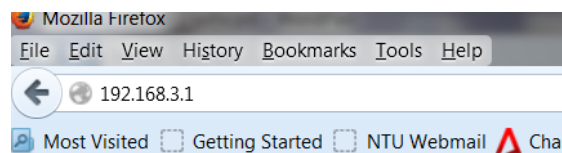
Use the Notebook's Wifi connection to scan and connect to the RPi SSID, and upon connection, a dynamically generated IP address will be assigned by the DHCP just set up.



At this juncture, you should be able to SSH to RPi using the Wifi IP address too.



If you have installed the Apache on RPi, use the browser and key in the RPi Wifi's IP address:



It works!

This is the default web page for this server.

The web server software is running but no content has been added, yet.

Notes on Raspberry Pi System Setup for MDP

C.4 – Setting up IP forwarding for gateway through eth0

So far, we have setup the RPi as a wireless access point with a DHCP that can provide IP address to device wirelessly connected to it. And this is what we will need for the MDP mobile robotic setup.

However, during the code and system development, we would like to also access the internet to download other packages and sample codes through the internet. Rather than having a separate internet connection, we can setup the RPi such that we can access the internet through its Ethernet LAN interface, but connects to the RPi through its Wifi access point using SSH.

IP forwarding is to be used such that packet received at the Wifi interface wlan0 will be forward to the Ethernet interface eth0 for internet access. (Remember that this is useful during the development work, but will not be needed for the eventually robotic setup since there won't be any wired connection to the robot)

- (i) Enable IP forwarding in the kernel using the following Linux command:

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

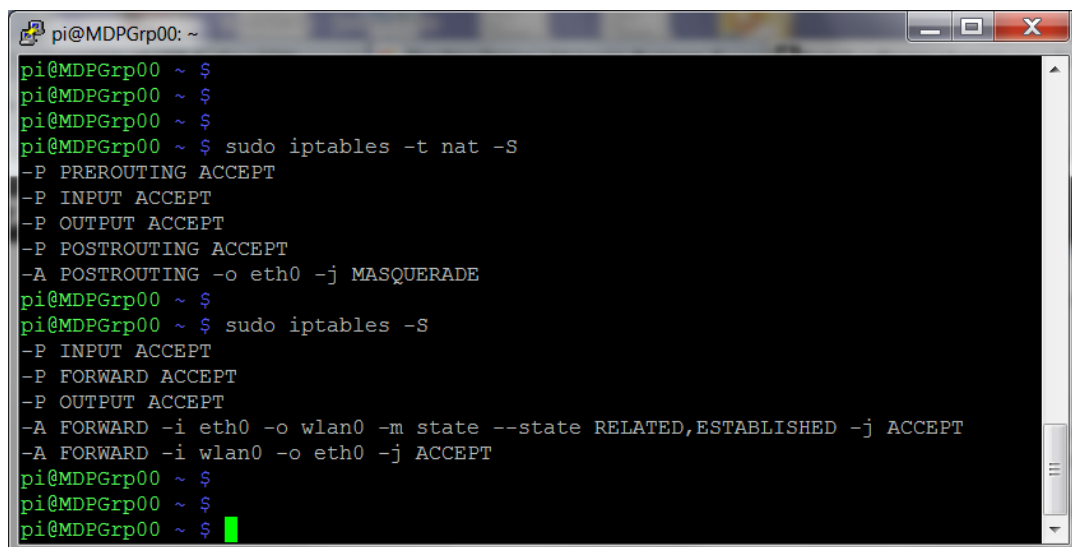
To make this setup automatically on boot, edit the file `/etc/sysctl.conf` and add the following line to the bottom of the file:

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

- (ii) Enable NAT in the kernel by executing the following three Linux commands:

```
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
```

Check the entries in the iptables to confirm the above settings, using command as shown.



```
pi@MDPGrp00: ~
pi@MDPGrp00 ~ $
pi@MDPGrp00 ~ $
pi@MDPGrp00 ~ $ sudo iptables -t nat -S
-P PREROUTING ACCEPT
-P INPUT ACCEPT
-P OUTPUT ACCEPT
-P POSTROUTING ACCEPT
-A POSTROUTING -o eth0 -j MASQUERADE
pi@MDPGrp00 ~ $
pi@MDPGrp00 ~ $ sudo iptables -S
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
-A FORWARD -i eth0 -o wlan0 -m state --state RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -i wlan0 -o eth0 -j ACCEPT
pi@MDPGrp00 ~ $
pi@MDPGrp00 ~ $
pi@MDPGrp00 ~ $
```

The IP forwarding should work now – which can be tested by accessing the Internet through a browser on the notebook that is Wifi linked to the RPi board.

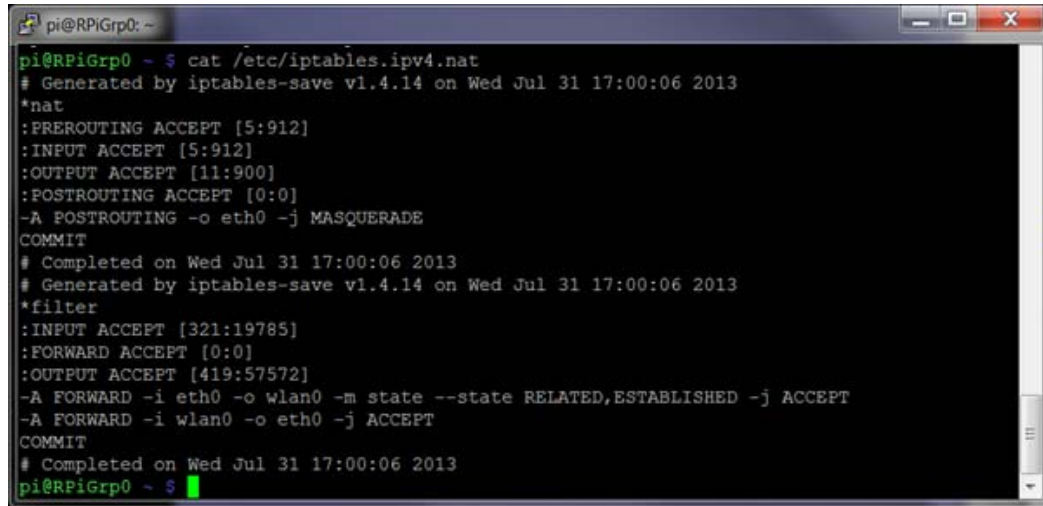
Notes on Raspberry Pi System Setup for MDP

- (iii) To set the iptables up automatically on boot, we will save the setting into a file first.

```
sudo sh -c "iptables-save > /etc/iptables.ipv4.nat"
```

Check that the file contains what were keyed in earlier by using the Linux command:

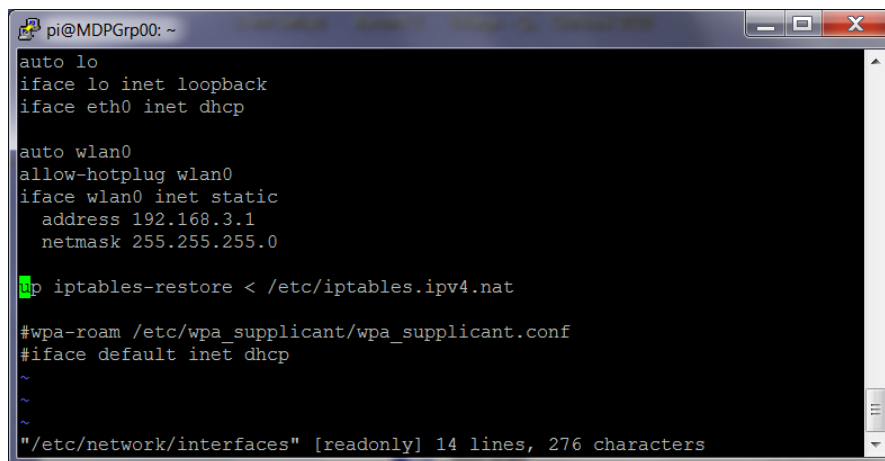
```
cat /etc/iptables.ipv4.nat
```



```
pi@RPiGrp0: ~
pi@RPiGrp0 ~ $ cat /etc/iptables.ipv4.nat
# Generated by iptables-save v1.4.14 on Wed Jul 31 17:00:06 2013
*nat
:PREROUTING ACCEPT [5:912]
:INPUT ACCEPT [5:912]
:OUTPUT ACCEPT [11:900]
:POSTROUTING ACCEPT [0:0]
-A POSTROUTING -o eth0 -j MASQUERADE
COMMIT
# Completed on Wed Jul 31 17:00:06 2013
# Generated by iptables-save v1.4.14 on Wed Jul 31 17:00:06 2013
*filter
:INPUT ACCEPT [321:19785]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [419:57572]
-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 Jul 31 17:00:06 2013
pi@RPiGrp0 ~ $
```

To use these setups upon reboot, edit the file `/etc/network/interfaces` and add the following to the end of the file:

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



```
pi@MDPGrp00: ~
auto lo
iface lo inet loopback
iface eth0 inet dhcp

auto wlan0
allow-hotplug wlan0
iface wlan0 inet static
    address 192.168.3.1
    netmask 255.255.255.0

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

#wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf
#iface default inet dhcp

~
~
~
"/etc/network/interfaces" [readonly] 14 lines, 276 characters
```

- (iv) Reboot the RPi board.
Once the system is up (Wifi dongle is blinking), SSH to the RPi board using its Wifi's IP address. The connection should show that it has internet access now.

From this point onward, the local monitor and keyboard are really not necessary now, as we can always SSH to the RPi board using the Wifi link, which is having a fixed IP address.



Notes on Raspberry Pi System Setup for MDP

D. Setting up of the Bluetooth link with Nexus 7 tablet.

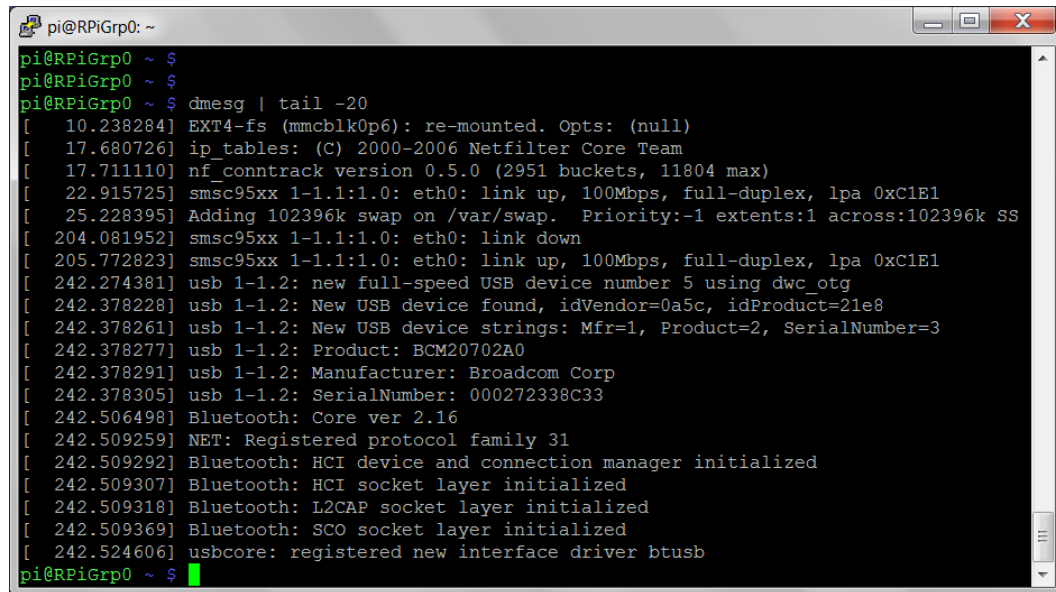
This section describes the setting and testing of the Bluetooth wireless interface between RPi and N7 tablet.

(Note, due to power consumption requirement, the Wifi dongle should be disconnected from the RPi, and the RPi should be accessed through the wired LAN network using SSH)

- (i) Plug the BT dongle into one of the USB port on RPi.

The dongle should be detected by the Raspbian: in the SSH console, type the Linux command to check:

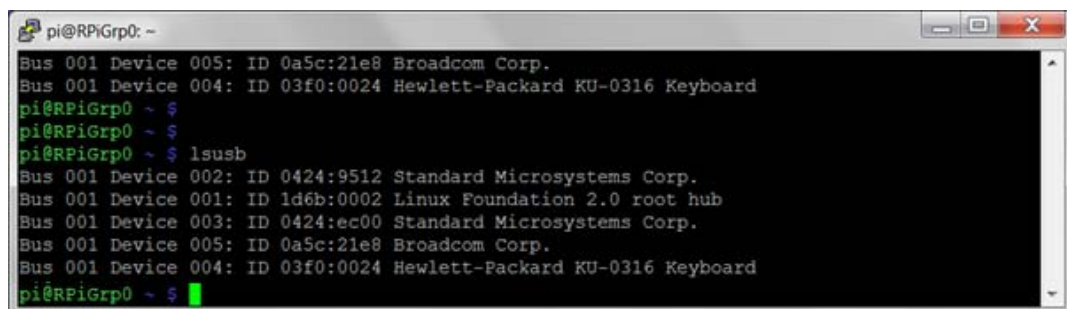
dmesg | tail -20



```
pi@RPiGrp0: ~
pi@RPiGrp0 ~ $
pi@RPiGrp0 ~ $ dmesg | tail -20
[ 10.238284] EXT4-fs (mmcblk0p6): re-mounted. Opts: (null)
[ 17.680726] ip_tables: (C) 2000-2006 Netfilter Core Team
[ 17.711110] nf_conntrack version 0.5.0 (2951 buckets, 11804 max)
[ 22.915725] smsc95xx 1-1.1:1.0: eth0: link up, 100Mbps, full-duplex, lpa 0xC1E1
[ 25.228395] Adding 102396k swap on /var/swap. Priority:-1 extents:1 across:102396k SS
[ 204.081952] smsc95xx 1-1.1:1.0: eth0: link down
[ 205.772823] smsc95xx 1-1.1:1.0: eth0: link up, 100Mbps, full-duplex, lpa 0xC1E1
[ 242.274381] usb 1-1.2: new full-speed USB device number 5 using dwc_otg
[ 242.378228] usb 1-1.2: New USB device found, idVendor=0a5c, idProduct=21e8
[ 242.378261] usb 1-1.2: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[ 242.378277] usb 1-1.2: Product: BCM20702A0
[ 242.378291] usb 1-1.2: Manufacturer: Broadcom Corp
[ 242.378305] usb 1-1.2: SerialNumber: 000272338C33
[ 242.506498] Bluetooth: Core ver 2.16
[ 242.509259] NET: Registered protocol family 31
[ 242.509292] Bluetooth: HCI device and connection manager initialized
[ 242.509307] Bluetooth: HCI socket layer initialized
[ 242.509318] Bluetooth: L2CAP socket layer initialized
[ 242.509369] Bluetooth: SCO socket layer initialized
[ 242.524606] usbcore: registered new interface driver btusb
pi@RPiGrp0 ~ $
```

You can also use the USB command: **lsusb**

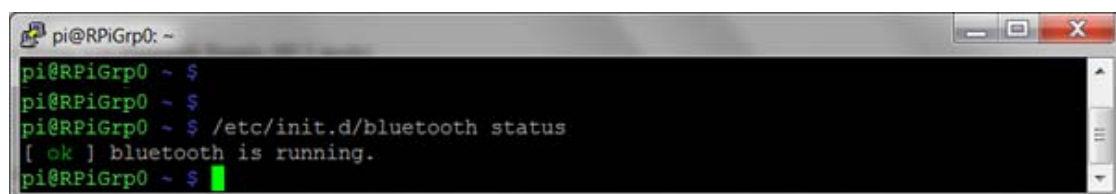
It should also shows that the BT dongle is detected as Broadcom corp.



```
pi@RPiGrp0: ~
pi@RPiGrp0 ~ $
pi@RPiGrp0 ~ $ lsusb
Bus 001 Device 005: ID 0a5c:21e8 Broadcom Corp.
Bus 001 Device 004: ID 03f0:0024 Hewlett-Packard KU-0316 Keyboard
pi@RPiGrp0 ~ $
pi@RPiGrp0 ~ $ lsusb
Bus 001 Device 002: ID 0424:9512 Standard Microsystems Corp.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 003: ID 0424:ec00 Standard Microsystems Corp.
Bus 001 Device 005: ID 0a5c:21e8 Broadcom Corp.
Bus 001 Device 004: ID 03f0:0024 Hewlett-Packard KU-0316 Keyboard
pi@RPiGrp0 ~ $
```

*Another command that can be used to check whether the BT is alive is to use the following command (But this command will only be available after we install the Bluetooth package - to be done in next section.)

/etc/init.d/bluetooth status



```
pi@RPiGrp0: ~
pi@RPiGrp0 ~ $
pi@RPiGrp0 ~ $ /etc/init.d/bluetooth status
[ ok ] bluetooth is running.
pi@RPiGrp0 ~ $
```

(This command can be used with other basic control functionality, by replacing 'status' with the following - 'start', 'stop', 'restart', 'force-reload')

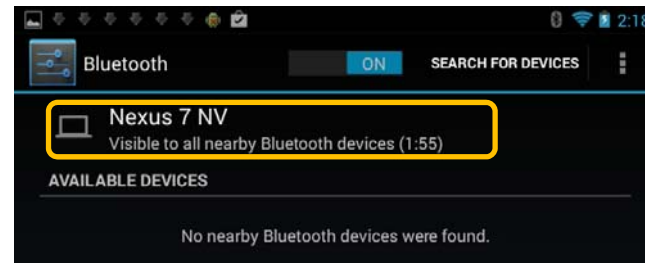
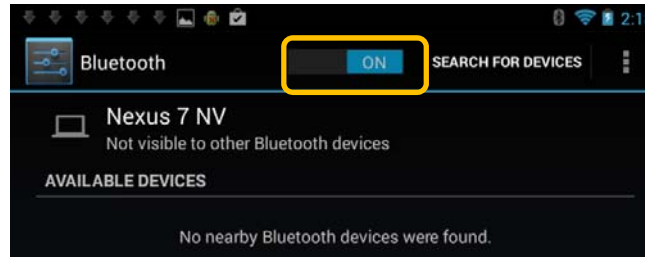
Notes on Raspberry Pi System Setup for MDP

- (ii) If everything looks OK so far, we will install other tools such that RPi can scan and connect to a BT master device (i.e. Nexus 7 tablet).

- (a) First install the Bluetooth package on the RPi using the following command:

```
sudo apt-get install bluetooth
```

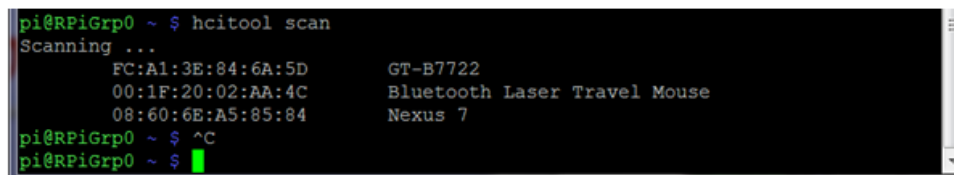
- (b) **On the Nexus 7 (N7)**, enable the Bluetooth (under the setting menu). Then sets it to '**Visible to all nearby Bluetooth devices**'. (Note this will remain on for 2 minutes, after that it will revert to non-visible.)



- (c) Back on the RPi, type the command:

```
hcitool scan
```

This will show any BT Master device in range of the RPi BT dongle

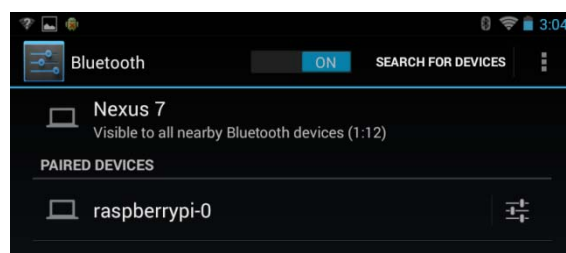
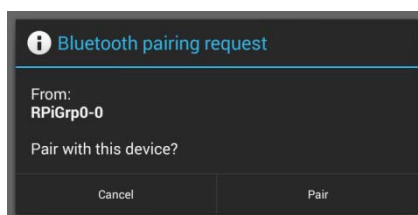


This shows that it has detected the N7 (as well as other devices such as my GT-B7722 feature phone and a BT mouse). Note the MAC address of N7: **08:60:6E:A5:85:84** (for my N7)

- (d) You can also 'ping' the N7 to check that they can communicate through the BT link. On the RPi, issue the following command (using YOUR N7's MAC address):

```
sudo l2ping -c 1 08:60:6E:A5:85:84
```

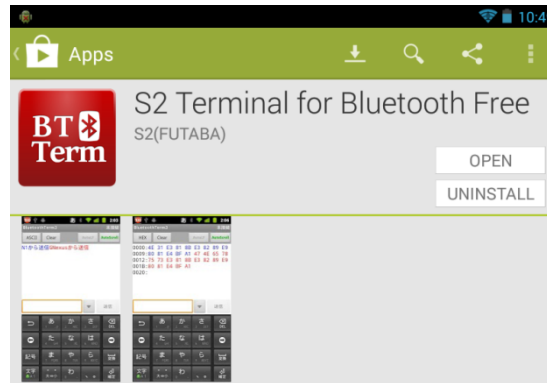
On N7, a "Bluetooth pairing request" message will appear. Select '**pair**' on N7.



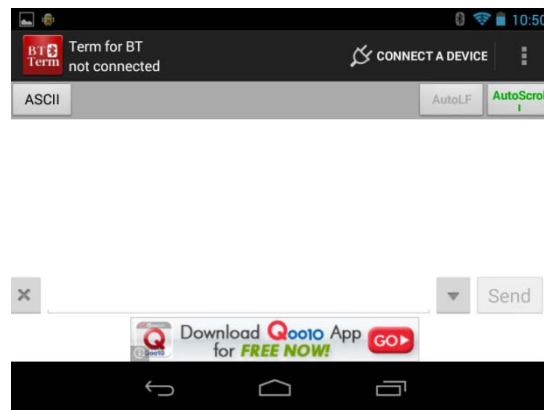
The RPi should also display a received response.

Notes on Raspberry Pi System Setup for MDP

- (iii) So, far, we have shown that the RPi and N7 can detect each other through the BT wireless link. Next we would check that data can be transferred between them, using the BT's serial port profile (SPP).
- (a) First we need to install an Android's terminal program based on BT on N7, which can operate using the SPP. We suggest using the free "S2 Terminal for Bluetooth", which can be easily installed through the Play Store on Nexus.



Once it is installed, start this BT-Term program on N7.



- (b) Back on RPi, run the command, using YOUR N7 MAC address.

`sdptool browse 08:60:6E:A5:85:84 | egrep "Service Name: |Channel:"`

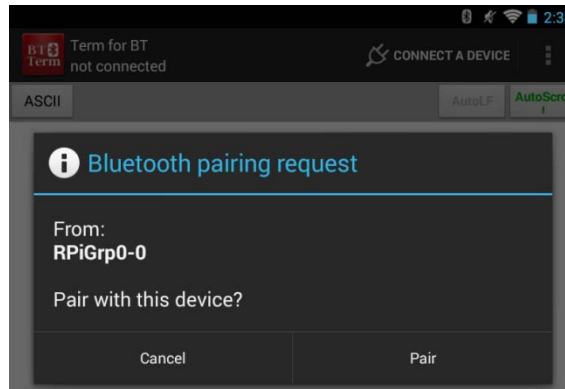
This will query the N7 for its available interfaces. Specifically we are looking for a channel that can support SPP. This should be available through Channel 4, indicated as "BluetoothInsecure".

```
pi@RPiGrp0: ~
pi@RPiGrp0 ~ $
pi@RPiGrp0 ~ $
pi@RPiGrp0 ~ $ sdptool browse 08:60:6E:A5:85:84 | egrep "Service Name: |Channel:"
Service Name: Headset Gateway
Channel: 2
Service Name: AV Remote Control Target
Service Name: Advanced Audio
Service Name: Android Network User
Service Name: OBEX Phonebook Access Server
Channel: 19
Service Name: OBEX Object Push
Channel: 12
Service Name: BluetoothSecure
Channel: 3
Service Name: BluetoothInsecure
Channel: 4
pi@RPiGrp0 ~ $
```

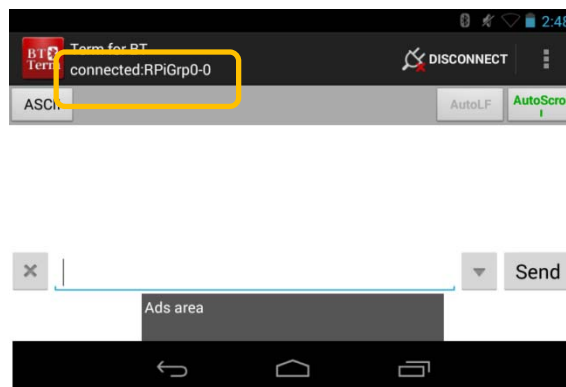
Note: If the BT-Term is not running on N7, this Channel 4 will not be available.

Notes on Raspberry Pi System Setup for MDP

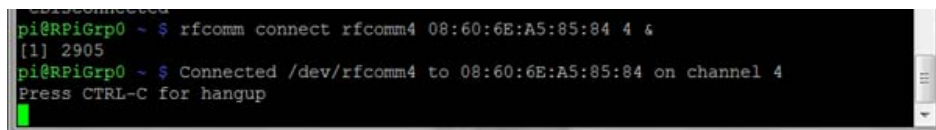
- (iv) Next we will try connecting RPi to N7 through channel 4 using the `rfcomm connect` command, and run it as a task in the background (i.e. using the `&` symbol).
- (a) Issue the following command on RPi
- ```
sudo rfcomm connect rfcomm4 08:60:6E:A5:85:84 4 &
```
- (b) A Bluetooth pairing request will be prompted on N7. Select “Pair”.



This will enable RPi to connect to the BT-Term (a **connected:hostname** will be shown at the top right corner of the BT Term program).



- (c) Back on RPi, a Connected message will also be shown.



A serial port is now established between the two devices, available on RPi through the file

**`/dev/rfcomm4`**

(All interfaces in Linux are through file I/O – as you will learn during the MDP)

- (v) We can now send messages between RPi and the N7 through **`/dev/rfcomm4`** BT serial port.
- (a) First press **Enter** to enter the prompt mode on RPi. Then issue the following command on RPi:

**`echo "Hi from RPi" > /dev/rfcomm4`**

## CE3004/CZ3004 MULTIDISCIPLINARY DESIGN PROJECT

### Notes on Raspberry Pi System Setup for MDP

The corresponding message will appear in the N7's BT-Term program screen.

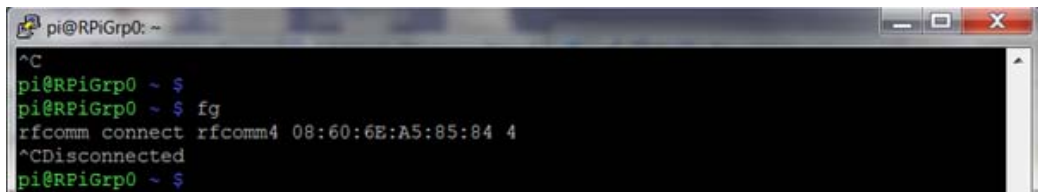


- (b) We can also send messages from N7 through the BT-Term to RPi. First get the RPi ready to display data in the 'file' /dev/rfcomm4 by typing the following command on RPi:
- ```
cat /dev/rfcomm4
```
- (c) Then on the N7 BT-Term type a message (such as "hi from n7"). The message should appear on the RPi console display too.



Type CTRL-C to get back into the prompt mode.

- (d) To terminate the connection, bring the background rfcomm task to the foreground by typing the command: **fg**, and then press **CTRL-C** to terminate the connection.



The BT Term program on N7 would also show '**not connected**' message now. (Alternatively the disconnection can be issued from the BT Term on N7.)

- (vi) Once the above are working, we can create the rfcomm4 and permanently store the RPi's BT setup by editing the following file:

```
sudo nano /etc/Bluetooth/rfcomm.conf
```

Add a configuration section with following entry.

```
# RFCOMM configuration file.
#
Rfcomm4 {
    # Automatically bind the device at startup
    bind yes;

    # Bluetooth MAC address of your N7
    device xx:xx:xx:xx:xx:xx;

    # RFCOMM channel for the connection
    channel 4;

    # Description of the connection
    comment "N7 BT Connection";
}
```

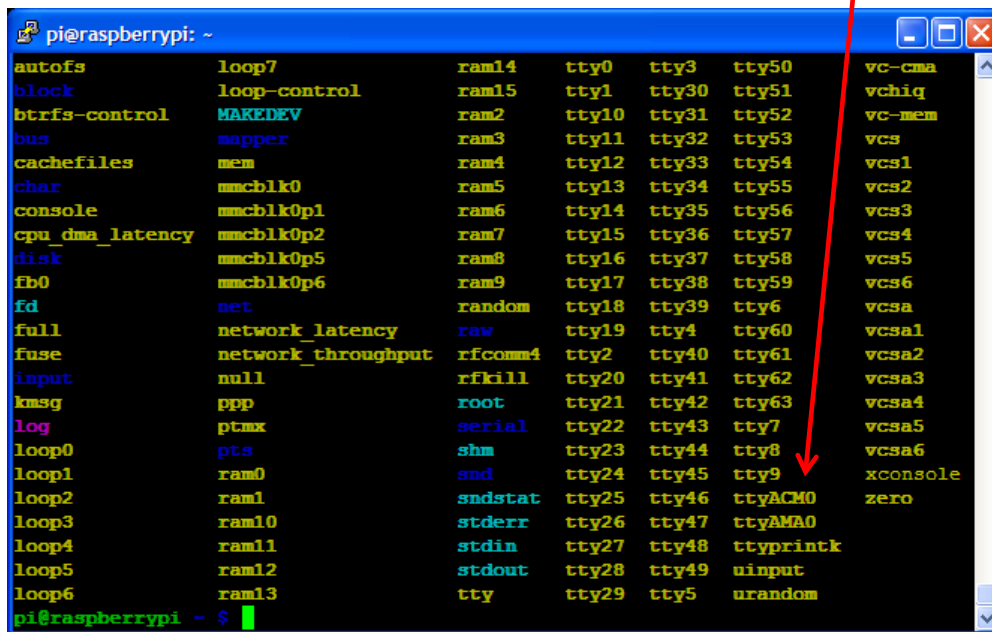
After rebooting the RPi, the /dev/rfcomm4 file will always be available and ready for connection.

Notes on Raspberry Pi System Setup for MDP

E. Setting up of the USB interface between RPi and Arduino board.

Connection between the RPi and Arduino will be through their USB port. Furthermore, the Arduino board will be powered by RPi through the same USB connection. Hence watch out for potential power supply problem as explained earlier. (However it is noted that on a fully assembled robotic setup powered by the 6V battery, the Arduino board will back-power the RPi, and the power issue should be resolved automatically.)

As the Arduino set its USB port to appear like a serial port based on the Abstract Control Model (ACM) serial interface that is supported by Raspbian, communication between the two devices can be similar done through the interface file automatically created on connection (which is typical `\dev\ttyACM0` in RPi).



```

pi@raspberrypi: ~
autofs          loop7          ram14          tty0           tty3           tty50          vc-cma
block           loop-control   ram15          tty1           tty30          tty51          vchiq
btrfs-control   MAKEDEV        ram2           tty10          tty31          tty52          vc-mem
bus             mapper         ram3           tty11          tty32          tty53          vcs
cachefiles      mem            ram4           tty12          tty33          tty54          vcs1
char            mmcblk0        ram5           tty13          tty34          tty55          vcs2
console         mmcblk0p1      ram6           tty14          tty35          tty56          vcs3
cpu_dma_latency mmcblk0p2      ram7           tty15          tty36          tty57          vcs4
disk            mmcblk0p5      ram8           tty16          tty37          tty58          vcs5
fb0             mmcblk0p6      ram9           tty17          tty38          tty59          vcs6
fd              net            random         tty18          tty39          tty6           vcsa
full            network_latency raw             tty19          tty4           tty60          vcsa1
fuse            network_throughput rfcomm4         tty2           tty40          tty61          vcsa2
input           null           rfkill         tty20          tty41          tty62          vcsa3
kmsg            ppp            root           tty21          tty42          tty63          vcsa4
log             ptmx           serial         tty22          tty43          tty7           vcsa5
loop0           pts            shm            tty23          tty44          tty8           vcsa6
loop1           ram0           snd            tty24          tty45          tty9           xconsole
loop2           ram1           sndstat        tty25          tty46          ttyACM0        zero
loop3           ram10          stderr         tty26          tty47          ttyAMA0
loop4           ram11          stdin          tty27          tty48          ttyprintk
loop5           ram12          stdout         tty28          tty49          uinput
loop6           ram13          tty            tty29          tty5           urandom
pi@raspberrypi ~ $

```

Data can be sent and displayed through the `ttyACM0` file, in the manner similar to the BT `rfcomm` connection. To demonstrate the data transfer, the Arduino board needs to run a corresponding program. (This will be left as an exercise for the student to work on).

F. Other packages

Other handy programs to have on the RPi are the following.

- (i) Apache web server on RPi to host webpage for access through Wifi.

```
sudo apt-get install apache2
```

Use a web browser on the connected notebook and key in the RPi's IP address <http://192.168.3.1>

- (ii) Samba Server on RPi to allow file transfer from the RPi and network computer (e.g. Win7)

```
sudo apt-get install samba samba-common-bin
```

(see http://elinux.org/R-Pi_NAS for further configuration detail)

With Samba server on RPi, user on another computer will be able to read and write to it as a networked drive which appears to be locally-attached but is actually attached to the RPi.

