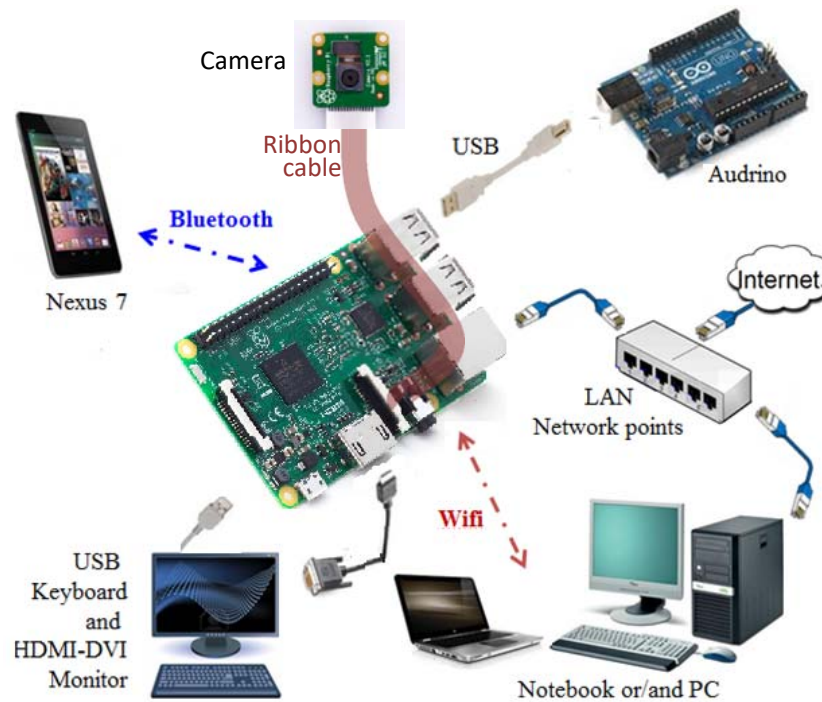


# CE3004/CZ3004 MULTIDISCIPLINARY DESIGN PROJECT

## Notes on Raspberry Pi System Setup for MDP



### System Overview

The Raspberry Pi (RPi 3) runs the Raspbian<sup>1</sup> 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 (AP) such that other computing devices can interface with it through Wifi link. However, during system development, its wired Ethernet connection will be connected to the LAN network point in the Lab, which provide 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 initial development work, such as to configure the network interface as well as downloading some of the software packages, the RPi can be connected to a local monitor<sup>2</sup> and keyboard through its USB ports such that it can be operated as a standalone computing platform. However, it will eventually be configured to be accessed 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.**

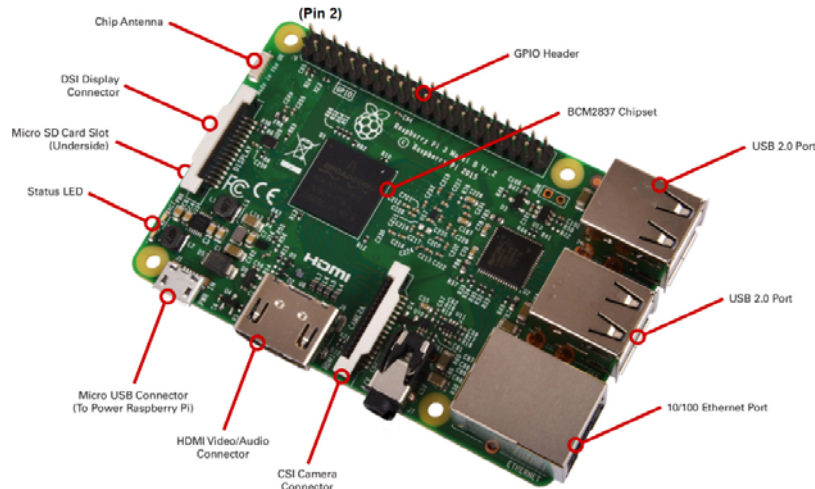
To save the files you create on RPi, you can install the Samba Server, which allow you to transfer the files over the network and store them on your own PC or thumbdrive.

### <sup>1</sup> 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.

# CE3004/CZ3004 MULTIDISCIPLINARY DESIGN PROJECT

## Notes on Raspberry Pi System Setup for MDP



The figure above shows the main components and interfaces on the Raspberry Pi 3 development board provided for this project. Its main processor is based on a 64-bit quad-core ARM Cortex A53 CPU. The RPi board was originally developed for teaching/learning computer programming purposes, but has since been widely used in many practical embedded applications.

### <sup>2</sup> Local monitor

The RPi will need to be connected to the Internet most of the time during the development work. However, before the Wifi AP<sup>54</sup> is up and running, the RPi will be assigned a dynamic IP address by the DHCP server used in the SCSE Lab's LAN. However, you need to know the <sup>3</sup>IP address in order to use the remote SSH through the network. Hence it is likely that you will need to connect the local monitor in order to observe the IP<sup>3</sup> allocated by the DHCP server during the bootup.

### <sup>3</sup> IP address

If the IP address is not shown in the bootup message, you could enable it through the `/etc/rc.local` script file (which may be already 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 a case, you will need the local keyboard and monitor to login and check by using the `ifconfig eth0` command.

### <sup>4</sup> Wireless Access Point and Internet gateway

The most convenient setup is to not connect the local monitor (and keyboard), but be able to SSH RPi through the network. For this, you 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. You can then use IP forwarding to send the IP packets to its wired Ethernet, which is connected to the LAN (which in turn has a dynamic IP address allocated by NTU network) and form a gateway to the Internet.

### Aside: Power Supply

The RPi will power all devices through its USB port, which is in turn itself powered through its Micro USB connector. However, RPi contains an onboard polyfuse whose resistance will increase as it gets hot. This is used to protect the RPi by limiting the total current allowable (about 750mA maximum) for the RPi. Beyond this current level, the fuse'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. As such, check the voltage level of the RPi if you encounter intermittence problems during system operation that seems to recover by itself.

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. Enabling the Wifi interface
  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.

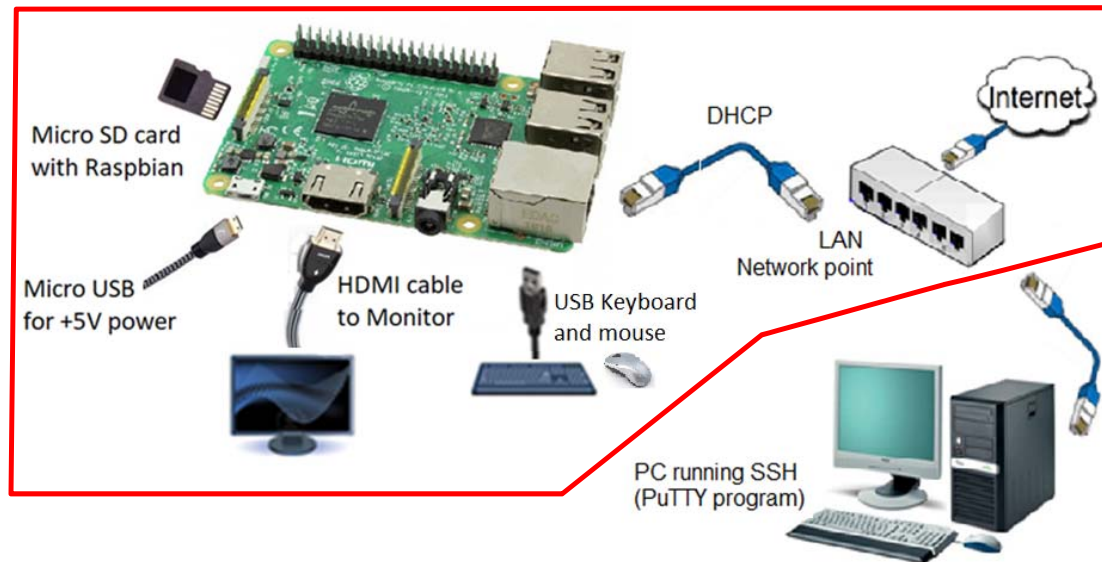
# CE3004/CZ3004 MULTIDISCIPLINARY DESIGN PROJECT

## Notes on Raspberry Pi System Setup for MDP

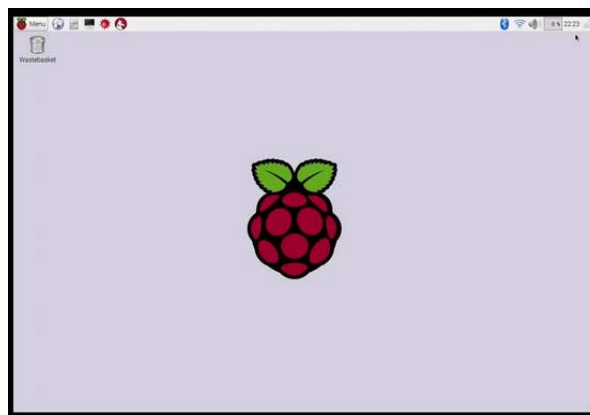
4. Configure the IP forwarding and NAT for gateway to Internet through the Ethernet connection.
- D. **Setting up the Bluetooth on RPi**
- E. Information on the **USB interface between RPi and Arduino board**
- F. Information on installation of **OpenCV software**
- G. Other useful packages to have on RPi
  1. Apache server to provide web-based GUI
  2. Samba server to transfer file over network

### A. Booting up the RPi with Raspbian

To begin with, the RPi should be connected to a monitor, keyboard and mouse, as well as the network point in the Lab for internet access as shown below.



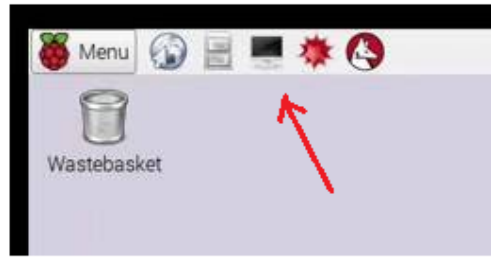
- (i) If your RPi has not been used before, the Micro SD card will contain an original Raspbian software that you can use 'straight out of the box'. But if it has been used before (by earlier MDP student), it should be re-formatted and installed with a new copy of the Raspbian - **See instruction in Appendix on how this should be done (recommended).**
- (ii) Once your Micro SD card has been installed with the new Raspbian O/S, insert the card into RPi 3 board (beneath the board), apply power to the RPi through the micro USB cable. The RPi should boot up and display series messages on the local monitor. Eventually it will boot into a GUI screen as shown below.



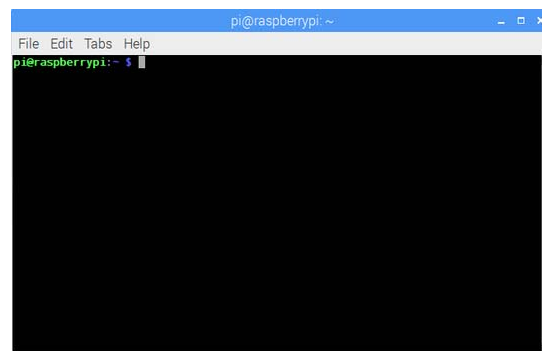
# CE3004/CZ3004 MULTIDISCIPLINARY DESIGN PROJECT

## Notes on Raspberry Pi System Setup for MDP

- (iii) However, you will find that many time it will be more convenient to use the RPi in the Command Line terminal mode (also known as console) for development work. To do so, click the black icon showing a console on the top left-hand corner of the display.



This will open a terminal window - the command line console, waiting for user to enter the necessary command.



### 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 eth0** to check whether an IP address has been assigned to the RPi. (It should be if the connection to the LAN network point is done properly.)

```
pi@MDPGrp00 ~ $ ifconfig eth0
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)
```

This IP address 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 note<sup>3</sup>.

You can also check whether your RPi has connection to the 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)

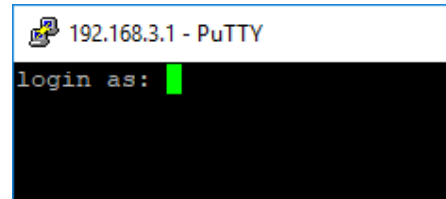
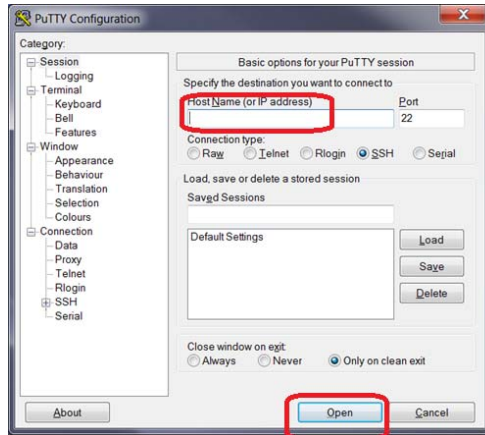
Note: in newer version of Raspbian, the Ethernet port Network interface may appear with name formed from a prefix en followed by x (indicating MAC) and its MAC address e.g. enx827eb123456.

# CE3004/CZ3004 MULTIDISCIPLINARY DESIGN PROJECT

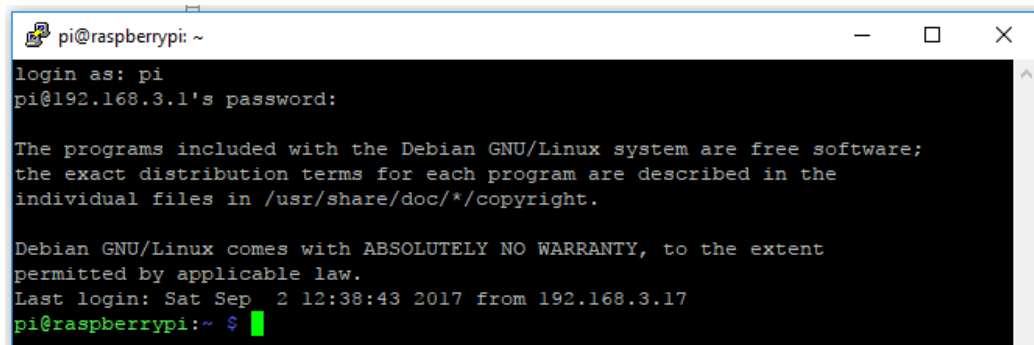
## Notes on Raspberry Pi System Setup for MDP

- (ii) Once the eth0's 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.

Run the PuTTY program and configure it with the RPi's IP address. Click Open. A console screen will appear on the remote PC . Log in using the default username and password (i.e. **pi** and **raspberrypi**).



- (iii) You should now have remote network access to the RPi through the LAN using the desktop PC (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)



Note: While the local keyboard/monitor/mouse 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<sup>3,4</sup> 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 setup<sup>5</sup> eventually, as you will achieve through this guide.

From this point onward, you will be using the remote computer to complete the procedures for the rest of this guide.

- (iv) Before you proceed further, it is better at this point to prepare the system for the subsequent configurations.

First execute the following command:

```
sudo apt-get update
```

This is to update the program package lists (from a server on the internet) such that you will always install the most recent version of the program when using the **apt-get** command.



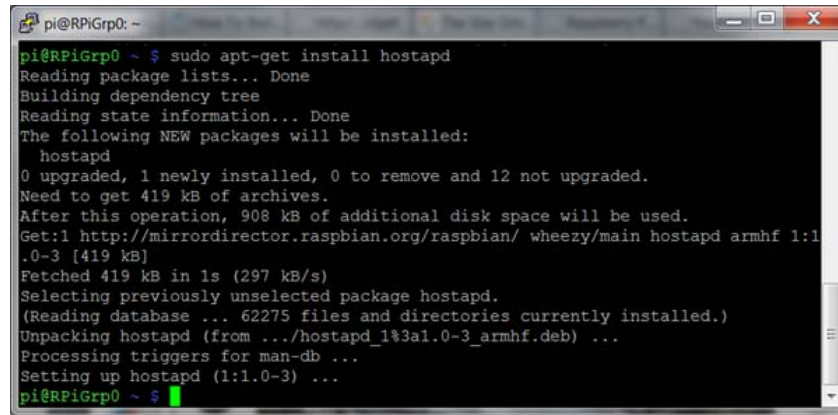
# CE3004/CZ3004 MULTIDISCIPLINARY DESIGN PROJECT

## Notes on Raspberry Pi System Setup for MDP

Next, you will download and install two programs ('packages' in Linux) that are to be used later in the setup, by issuing the following Linux commands in the SSH client console.

- (a) First is the Host Access Point package **hostapd**:

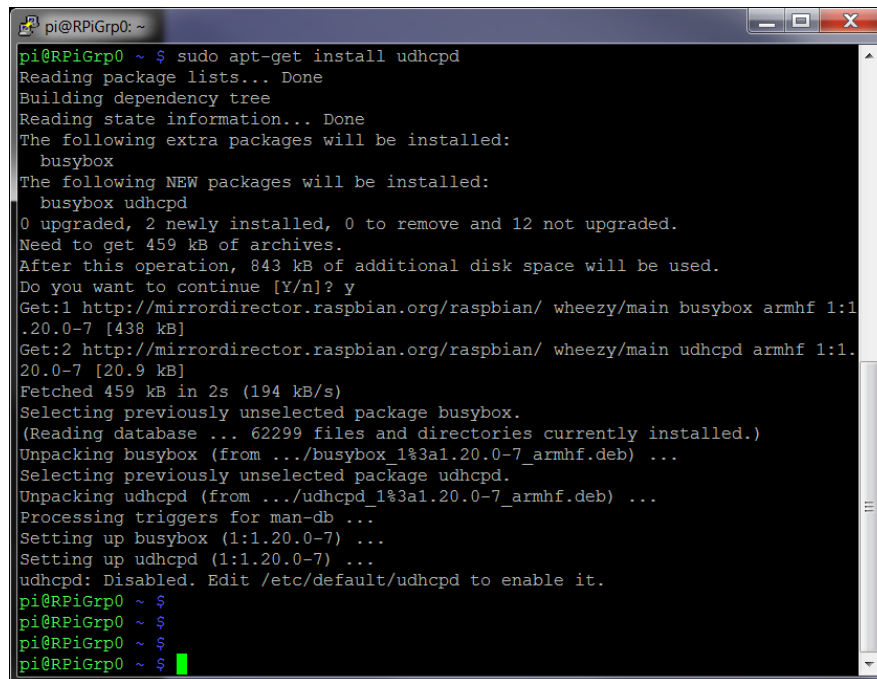
```
sudo apt-get install hostapd
```

A terminal window titled 'pi@RPiGrp0: ~' showing the command 'sudo apt-get install hostapd' being executed. The output shows the package lists being read, the dependency tree being built, and the state information being read. It then lists 'hostapd' as a new package to be installed. The disk space requirements are shown: 419 kB of archives and 908 kB of additional disk space. The package is fetched from the mirror and installed. The terminal output is as follows:

```
pi@RPiGrp0 ~ $ sudo apt-get install hostapd
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  hostapd
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 hostapd armhf 1:1.0-3 [419 kB]
Fetched 419 kB in 1s (297 kB/s)
Selecting previously unselected package hostapd.
(Reading database ... 62275 files and directories currently installed.)
Unpacking hostapd (from .../hostapd_1%3a1.0-3_armhf.deb) ...
Processing triggers for man-db ...
Setting up hostapd (1:1.0-3) ...
pi@RPiGrp0 ~ $
```

- (b) Next is the DHCP server package. In here, you will use the **udhcpd** (a popular alternative is the **dnsmasq**)

```
sudo apt-get install udhcpd
```

A terminal window titled 'pi@RPiGrp0: ~' showing the command 'sudo apt-get install udhcpd' being executed. The output shows the package lists being read, the dependency tree being built, and the state information being read. It then lists 'busybox' and 'udhcpd' as extra packages to be installed. The disk space requirements are shown: 459 kB of archives and 843 kB of additional disk space. The packages are fetched from the mirror and installed. The terminal output is as follows:

```
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 ~ $
```

Although these two packages are now installed in the RPi system, they need to be further configured and enabled before they can be activated in the system.

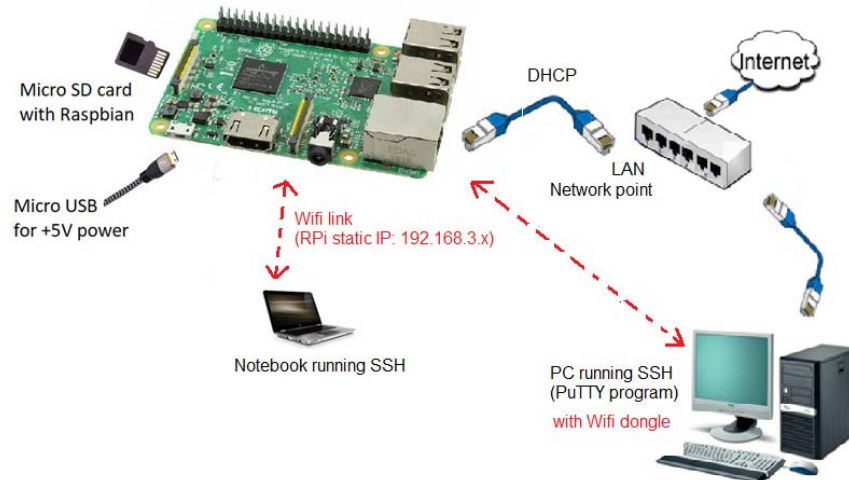
Another useful utility is the program to stop the RPi from going into sleeping mode: **xscreensaver**

**Aside:** There are tons of information on the internet related to the Raspberry Pi. Hence use Google to search for relevant information when you encounter a problem, or look for solutions.

### 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.

If you are allocated a desktop PC (instead of the notebook) in the Lab that do not have built-in Wifi, a Wifi dongle (e.g. Edimax) should be inserted 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 or mobile phones (which would have come with Wifi) to verify the various steps during the following setup.



#### C.1 – Setting up hostapd

The main program needed for setting up the RPi as a wireless access point is **hostapd**, which should have been downloaded and installed earlier. This is a user space program (as opposed to kernel space program in Linux) that will run in the background (known as **daemon**) and provide the wireless **host** access **point** service (google to find out more detail about this program).

- (i) If the hostapd is successfully installed earlier, a **hostapd** binary file would be created in a `/usr/sbin/` subdirectory as follows:

**`/usr/sbin/hostapd`**

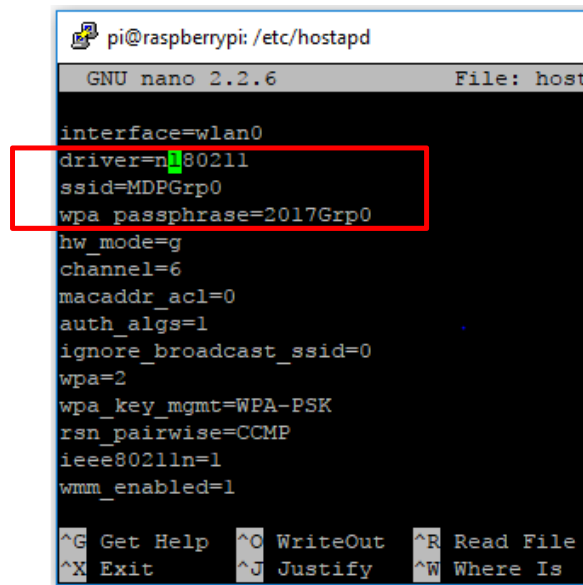
together with other necessary files, such as those found in the subdirectory `/etc/hostapd/`

- (ii) First create a configuration file **hostapd.conf**, which is needed to configure hostapd. (You can use the Linux command line text editor program, **nano** to edit it, using the command as shown below. Google 'Linux nano' for detail of the programs usage)

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

# CE3004/CZ3004 MULTIDISCIPLINARY DESIGN PROJECT

## Notes on Raspberry Pi System Setup for MDP



```
pi@raspberrypi: /etc/hostapd
GNU nano 2.2.6 File: hostapd.conf

interface=wlan0
driver=nl80211
ssid=MDPGrp0
wpa_passphrase=2017Grp0
hw_mode=g
channel=6
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_key_mgmt=WPA-PSK
rsn_pairwise=CCMP
ieee80211n=1
wmm_enabled=1

^G Get Help ^O WriteOut ^R Read File
^X Exit ^J Justify ^W Where Is
```

The configuration includes the driver (should be **NL80211**, 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:

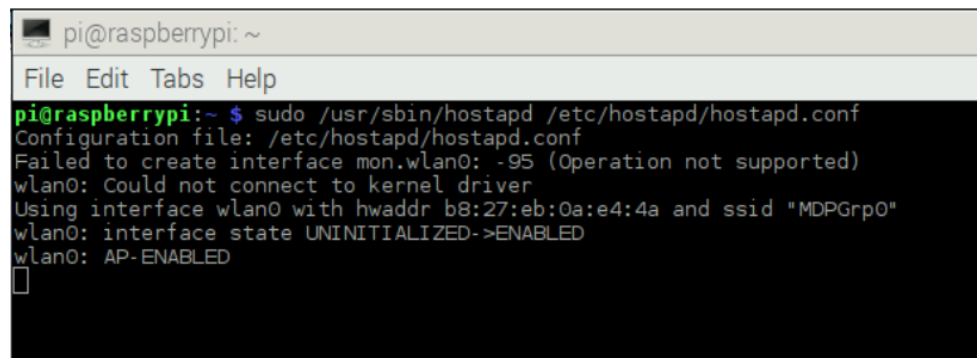
Group 1 should set the **ssid=MDPGrp1**

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

- (v) To verify the setup, execute the command

```
sudo /usr/sbin/hostapd /etc/hostapd/hostapd.conf
```

A series of messages should appear in the console, showing the access point is enabled.



```
pi@raspberrypi: ~
File Edit Tabs Help
pi@raspberrypi:~ $ sudo /usr/sbin/hostapd /etc/hostapd/hostapd.conf
Configuration file: /etc/hostapd/hostapd.conf
Failed to create interface mon.wlan0: -95 (Operation not supported)
wlan0: Could not connect to kernel driver
Using interface wlan0 with hwaddr b8:27:eb:0a:e4:4a and ssid "MDPGrp0"
wlan0: interface state UNINITIALIZED->ENABLED
wlan0: AP-ENABLED
```

Use your notebook/mobile phone wifi to scan for the network, the wifi ssid should appear on the screen if it is setup successfully.

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

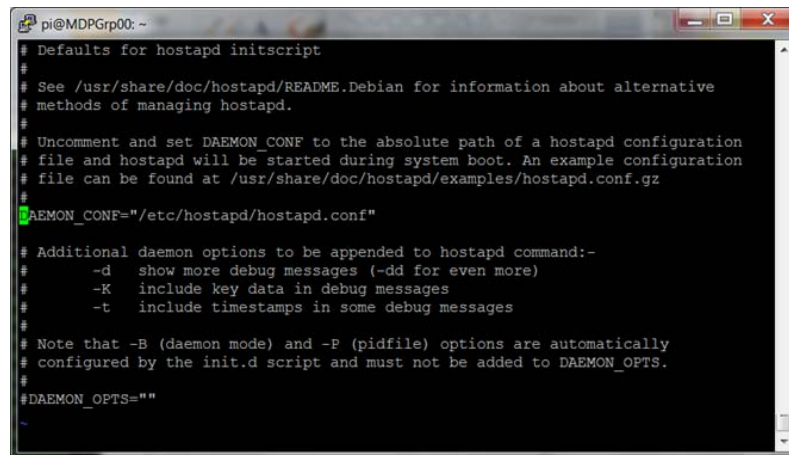
Change **#DAEMON\_CONF=""**

to **DAEMON\_CONF="/etc/hostapd/hostapd.conf"**



# CE3004/CZ3004 MULTIDISCIPLINARY DESIGN PROJECT

## Notes on Raspberry Pi System Setup for MDP



```
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=""
```

- (vi) To start the hostapd program, you can use the following command:

```
sudo service hostapd start
```

But before you continue, stop the access point program at this juncture.

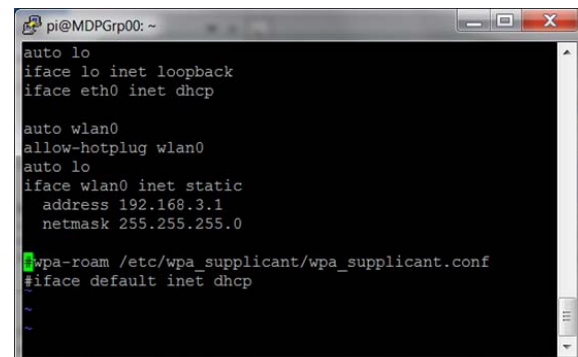
```
sudo service hostapd stop
```

### C.2 – Setting up the Wireless interface wlan0

Before you configure the DHCP program that you installed earlier, you will first assign a static IP address to the Wifi interface, which is indicated as wlan0 in Linux. (The wired Ethernet interface is normally 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 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

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)

*Note:* If you have difficulty establishing network connection for **eth0** using dhcp, try

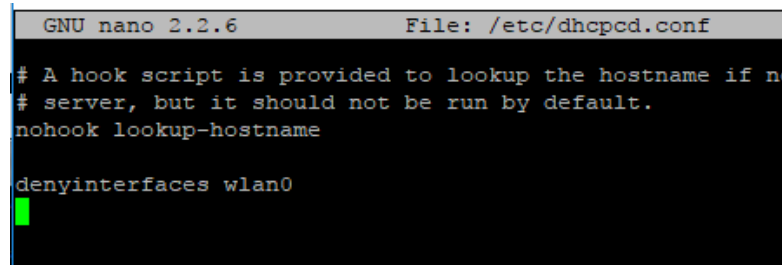
- connecting the Ethernet cable to the LAN network point before boot-up
- adding the line **auto eth0** before **iface eth0 inet dhcp**

## CE3004/CZ3004 MULTIDISCIPLINARY DESIGN PROJECT

### Notes on Raspberry Pi System Setup for MDP

- (ii) In the new version of Raspbian (after Aug 2017), the network will also depend on the settings used in the file `/etc/dhcpd.conf`. To disable its effect on the Wifi setting, add the following line to the end of this file

`denyinterface wlan0`



```
GNU nano 2.2.6 File: /etc/dhcpd.conf

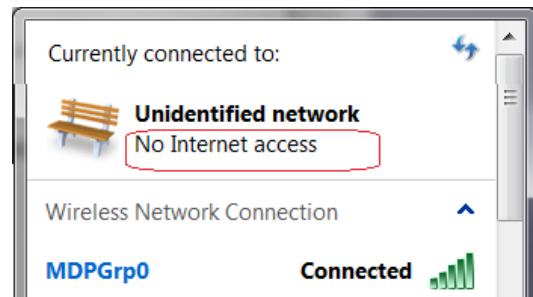
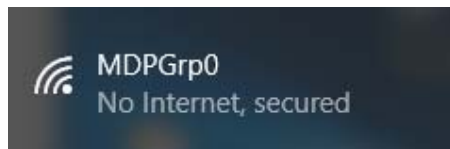
# A hook script is provided to lookup the hostname if no
# server, but it should not be run by default.
nohook lookup-hostname

denyinterfaces wlan0
```

- (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. (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, and also without any Internet connection.



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

### 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 you will use here is the **udhcpd** (which has been downloaded and installed earlier), but you can use other appropriate one (e.g. dnsmasq) as you prefer – they are available through many references on RPi 3 sites on internet.

- (i) After the udhcpd package is installed earlier, there is a new file created: `/etc/udhcpd.conf`. The appropriate lines in the file need to be edited to as shown in the next page (using command `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)

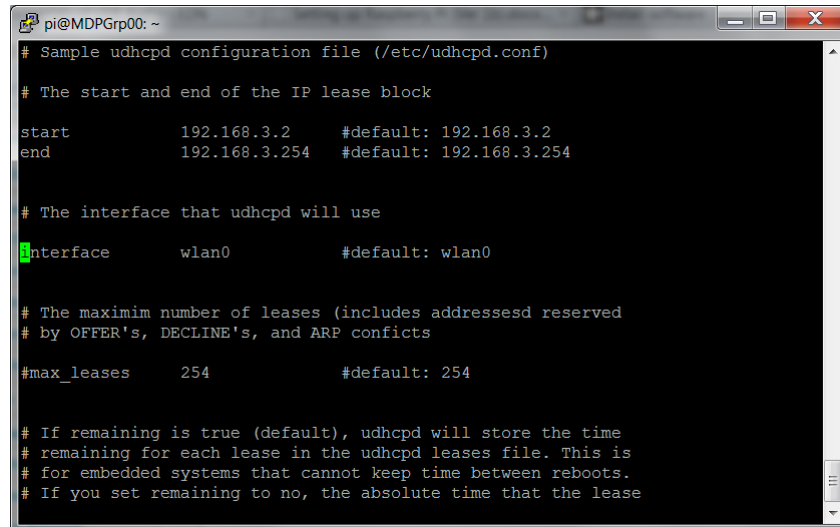
# CE3004/CZ3004 MULTIDISCIPLINARY DESIGN PROJECT

## Notes on Raspberry Pi System Setup for MDP

```
# 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
```



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

# The start and end of the IP lease block

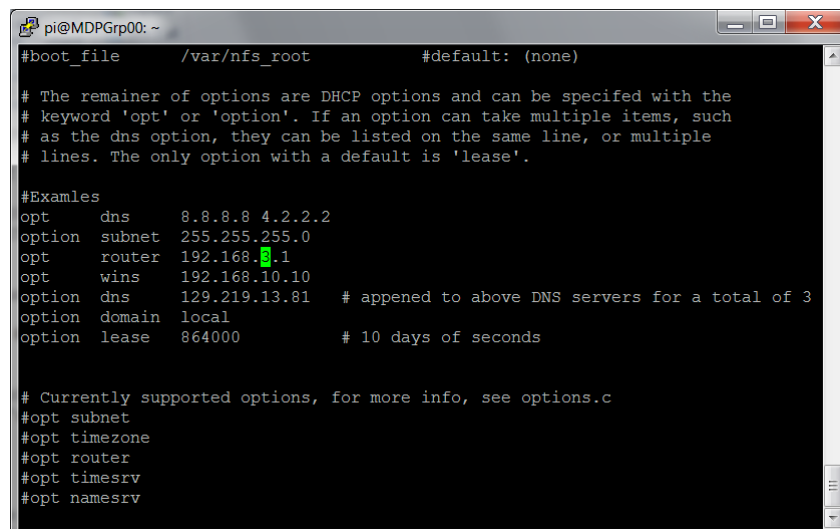
start      192.168.3.2      #default: 192.168.3.2
end        192.168.3.254   #default: 192.168.3.254

# 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
opt  subnet   255.255.255.0
opt  router   192.168.3.1
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
```

There is another file that 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"**

# CE3004/CZ3004 MULTIDISCIPLINARY DESIGN PROJECT

## Notes on Raspberry Pi System Setup for MDP

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

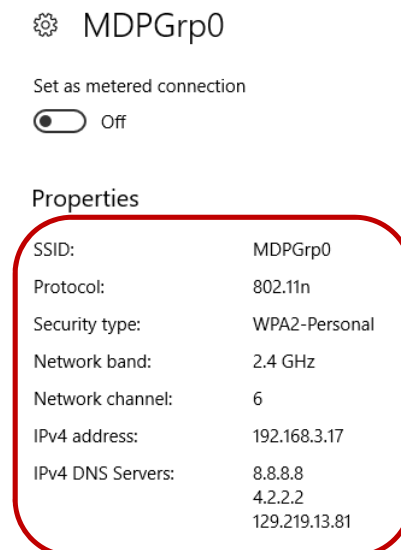
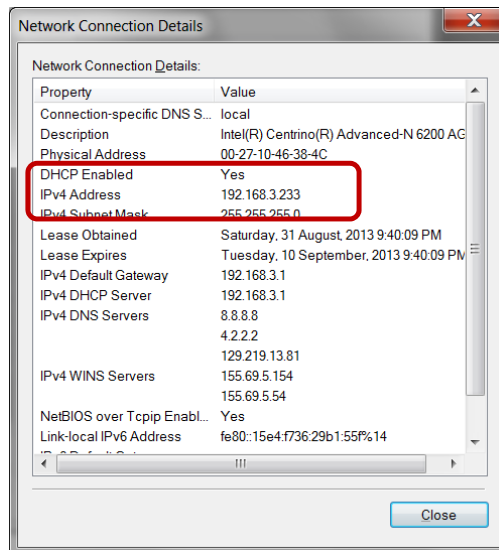
# Options to pass to busybox' udhcpd.
#
# -S    Log to syslog
# -f    run in foreground

DHCPD_OPTS="-S"
~
"/etc/default/udhcpd" [readonly] 9 lines, 165 characters
```

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

**sudo service udhcpd restart**

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, using your notebook, you should also be able to SSH to RPi using its Wifi IP address too (i.e. 192.168.3.1).

```
pi@raspberrypi: ~
login as: pi
pi@192.168.3.1's password:

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: Sat Sep  2 14:22:53 2017 from 192.168.3.17
pi@raspberrypi:~$
```

# CE3004/CZ3004 MULTIDISCIPLINARY DESIGN PROJECT

## Notes on Raspberry Pi System Setup for MDP

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

At this stage, you have setup the RPi as a wireless access point with a DHCP server that can provide IP address to devices wirelessly connected to it. And this is what you will need for the MDP mobile robotic setup.

However, during coding and system development, you would like to also access the internet to download other packages and sample codes through the internet. Rather than having a separate internet connection, you can setup the RPi such that internet can be accessed through its Ethernet LAN interface (connects to the LAN network point in the Lab) while you remote access the RPi through its Wifi interface using SSH.

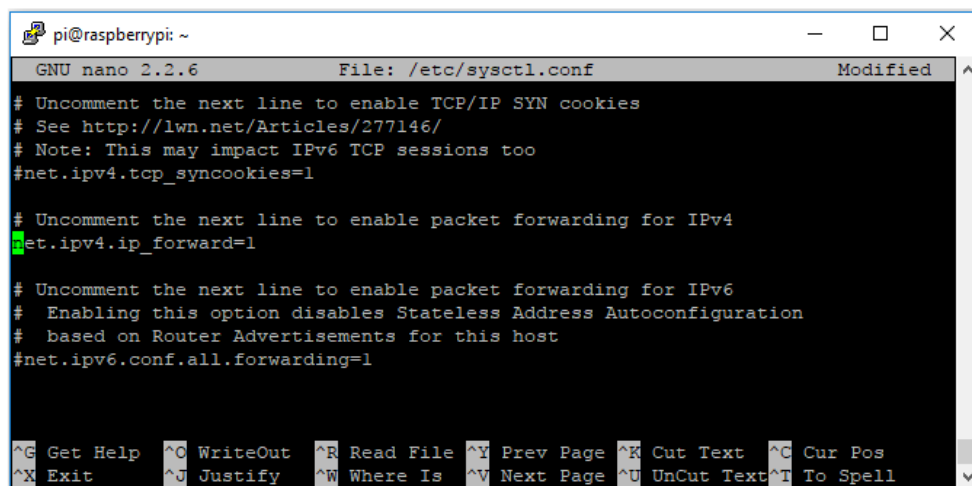
To do so, IP forwarding is to be used such that packet received at the Wifi interface wlan0 is forward to the Ethernet interface eth0 for internet access. (Remember that this is useful during the development work, but will not be needed for the eventually mobile robot setup since there won't be any wired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

- (i) To enable IP forwarding in the kernel, issue the following Linux command in the terminal:

```
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 uncomment the following line in the file:

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



```
pi@raspberrypi: ~
GNU nano 2.2.6 File: /etc/sysctl.conf Modified
# Uncomment the next line to enable TCP/IP SYN cookies
# See http://lwn.net/Articles/277146/
# Note: This may impact IPv6 TCP sessions too
#net.ipv4.tcp_syncookies=1

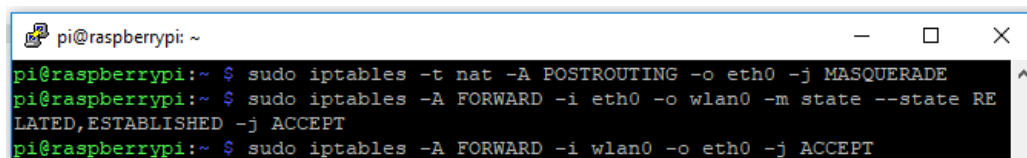
# Uncomment the next line to enable packet forwarding for IPv4
net.ipv4.ip_forward=1

# Uncomment the next line to enable packet forwarding for IPv6
# Enabling this option disables Stateless Address Autoconfiguration
# based on Router Advertisements for this host
#net.ipv6.conf.all.forwarding=1

^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell
```

- (ii) Next you will enable the NAT (Network Address Translation) feature by executing three iptables rules such that RPi will transfer packets that comes through wlan0 interface (i.e. Wifi) to the eth0 interface (i.e. Ethernet port) connected to the Internet.

```
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
```



```
pi@raspberrypi:~ $ sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
pi@raspberrypi:~ $ sudo iptables -A FORWARD -i eth0 -o wlan0 -m state --state RE
LATED,ESTABLISHED -j ACCEPT
pi@raspberrypi:~ $ sudo iptables -A FORWARD -i wlan0 -o eth0 -j ACCEPT
```

You can check the entries in the iptables to confirm the above settings, using the following command.

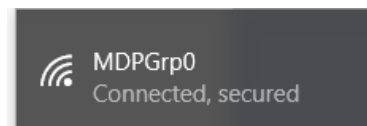
```
sudo iptables -t nat -S
```

# CE3004/CZ3004 MULTIDISCIPLINARY DESIGN PROJECT

## Notes on Raspberry Pi System Setup for MDP

```
pi@raspberrypi: ~  
pi@raspberrypi:~$ sudo iptables -t nat -S  
-P PREROUTING ACCEPT  
-P INPUT ACCEPT  
-P OUTPUT ACCEPT  
-P POSTROUTING ACCEPT  
-A POSTROUTING -o eth0 -j MASQUERADE  
pi@raspberrypi:~$ 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@raspberrypi:~$
```

The IP forwarding should work now. Connect the RPi to a LAN port using the Ethernet cable. The internet connection will be reflected on the notebook in its Wifi link status as “connected”:



You can test the Internet connection by running the browser on the notebook that is Wifi linked to the RPi board.

- iii) To keep the iptables rules automatically on boot-up, first save the setting into a file.

```
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@raspberrypi: ~  
pi@raspberrypi:~$ cat /etc/iptables.ipv4.nat  
# Generated by iptables-save v1.4.21 on Sun Sep  3 14:12:10 2017  
*filter  
:INPUT ACCEPT [391:62315]  
:FORWARD ACCEPT [0:0]  
:OUTPUT ACCEPT [96:11556]  
-A FORWARD -i eth0 -o wlan0 -m state --state RELATED,ESTABLISHED -j ACCEPT  
-A FORWARD -i wlan0 -o eth0 -j ACCEPT  
COMMIT  
# Completed on Sun Sep  3 14:12:10 2017  
# Generated by iptables-save v1.4.21 on Sun Sep  3 14:12:10 2017  
*nat  
:PREROUTING ACCEPT [257:22112]  
:INPUT ACCEPT [28:3158]  
:OUTPUT ACCEPT [30:2280]  
:POSTROUTING ACCEPT [0:0]  
-A POSTROUTING -o eth0 -j MASQUERADE  
COMMIT  
# Completed on Sun Sep  3 14:12:10 2017  
pi@raspberrypi:~$
```

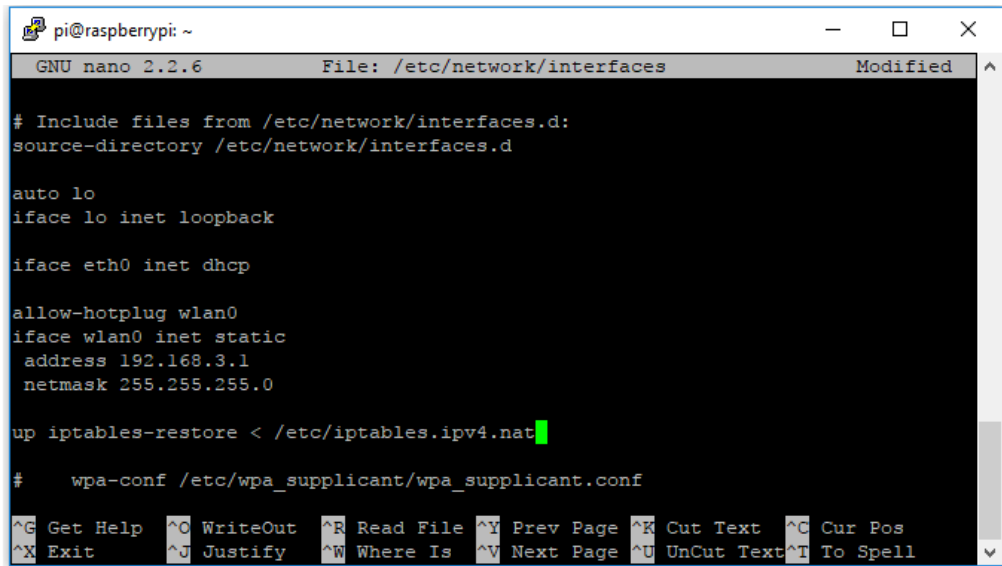
To apply these setup rules upon reboot, edit the file **/etc/network/interfaces** and add the following to the end of the file:

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



# CE3004/CZ3004 MULTIDISCIPLINARY DESIGN PROJECT

## Notes on Raspberry Pi System Setup for MDP



```
pi@raspberrypi: ~  
GNU nano 2.2.6 File: /etc/network/interfaces Modified  
  
# Include files from /etc/network/interfaces.d:  
source-directory /etc/network/interfaces.d  
  
auto lo  
iface lo inet loopback  
  
iface eth0 inet dhcp  
  
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-conf /etc/wpa_supplicant/wpa_supplicant.conf  
  
^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos  
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell
```

- (iv) Reboot the RPi board: **sudo shutdown -r 0**

Once the wifi network is detected (scan for the wifi network), connect to it and SSH to the RPi board using its Wifi's IP address. The connection should show that it has internet access now.

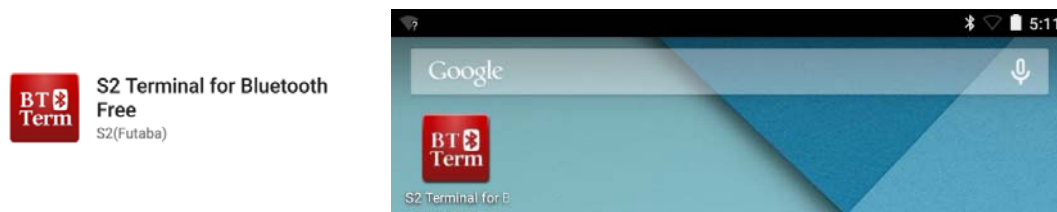
From this point onward, the local monitor and keyboard are really not necessary anymore, as you can always SSH to the RPi board using the Wifi link, which is having a fixed IP address.

### D. Setting up the RPi Bluetooth link with Nexus 7 Tablet.

RPi3 comes with a built-in Bluetooth transceiver which can be used to connect to other Bluetooth device such as the Nexus 7 (N7) tablet and smart phone. In here, you will try to establish a Bluetooth connection between the RPi3 and N7 tablet, communicating using the Serial Port profile (SPP).

#### D.1 Preparation work

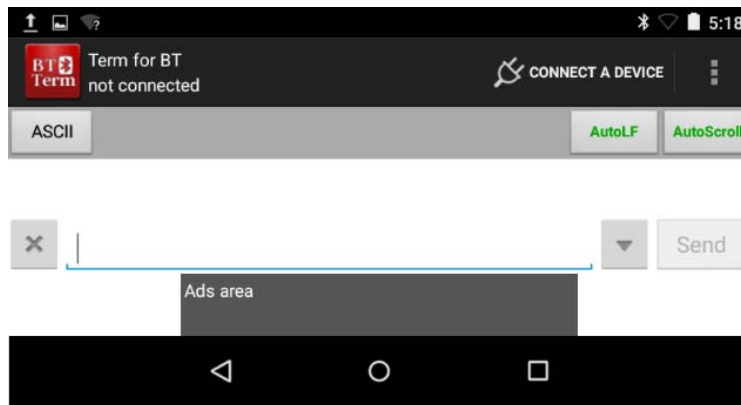
- (a) To be able to verify the proper configuration of the RPi3 bluetooth connection, you need a bluetooth application (APP) to run on the Nexus 7 that utilizes the SPP. (You will eventually develop such an APP for your MDP system.) For the initial setup verification purpose, you will first install a free APP (from the Play Store) on the Nexus 7 tablet. In this document, the APP used for the setup verification is the BT-Term.



Download and install the BT-Term APP through the Google Play Store. Test run the APP on the N7 tablet, and it should indicate that it is ready, but waiting to connect to a device as shown in the following screenshot. Close the APP as you will need to pair its Bluetooth with the RPi first.

# CE3004/CZ3004 MULTIDISCIPLINARY DESIGN PROJECT

## Notes on Raspberry Pi System Setup for MDP



- (b) For successful interfacing with N7, the RPi3 bluetooth daemon process need to be running in the compatible mode. This can be done through the file `/etc/systemd/system/dbus-org.bluez.service` as follows.

```
sudo nano /etc/systemd/system/dbus-org.bluez.service
```

Add `-C` at the end of the `ExecStart=` line.

```
pi@raspberrypi: ~  
GNU nano 2.2.6 File: /etc/systemd/system/dbus-org.bluez.service Modified  
[Unit]  
Description=Bluetooth service  
Documentation=man:bluetoothd(8)  
  
[Service]  
Type=dbus  
BusName=org.bluez  
ExecStart=/usr/lib/bluetooth/bluetoothd -C  
NotifyAccess=main  
#WatchdogSec=10  
#Restart=on-failure  
CapabilityBoundingSet=CAP_NET_ADMIN CAP_NET_BIND_SERVICE  
LimitNPROC=1  
  
[Install]  
WantedBy=bluetooth.target  
Alias=dbus-org.bluez.service  
  
^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos  
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell
```

Then restart the RPi: `sudo shutdown -r 0`.

### D.2 Checking the RPi 3 Bluetooth status

It is likely that the RPi 3 Bluetooth drivers are already up and running upon boot-up (i.e. by default). You can check its drivers by typing through the SSH terminal, the command: `lsmod`. There should be three drivers as shown above.

```
pi@raspberrypi:~$ lsmod  
Module      Size  Used by  
rfcomm      33586   7  
bnep        10336   2  
hci_uart    13533   1  
btbcm       4196   1 hci_uart  
bluetooth   317981  27 bnep,btbcm,hci_uart,rfcomm
```

# CE3004/CZ3004 MULTIDISCIPLINARY DESIGN PROJECT

## Notes on Raspberry Pi System Setup for MDP

Next, check the status of the Bluetooth interface using the command:

```
systemctl status bluetooth
```

```
pi@raspberrypi:~ $ systemctl status bluetooth
● bluetooth.service - Bluetooth service
   Loaded: loaded (/lib/systemd/system/bluetooth.service; enabled)
   Active: active (running) since Sun 2017-09-03 22:41:00 UTC; 2h 19min ago
     Docs: man:bluetoothd(8)
   Process: 639 ExecStartPost=/usr/bin/sdptool add --channel=4 SP (code=exited, status=0/SUCCESS)
   Main PID: 635 (bluetoothd)
     Status: "Running"
    CGroup: /system.slice/bluetooth.service
           └─635 /usr/lib/bluetooth/bluetoothd -C

pi@raspberrypi:~ $
```

It should indicate that the Bluetooth service status is “Running”. (Otherwise, issue the command `sudo service bluetooth start` )

### D.3 Configuring the RPi Bluetooth interface with N7

Next, you will be using the `hcitool` command to configure the Bluetooth interface. (Note: At this stage, you might want to also run the `bluetoothctl` utility on another terminal that can monitor the bluetooth activities – see D.7)

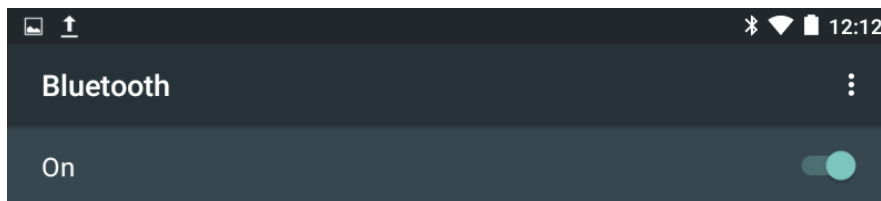
Type `hcitool dev` and this will show the hardware device address of the RPi.

```
pi@raspberrypi:~ $ hcitool dev
Devices:
        hci0      43:43:A1:12:1F:AC
pi@raspberrypi:~ $
```

Then scan for active discoverable Bluetooth devices nearby using the command: `sudo hcitool scan`

```
pi@raspberrypi: ~
pi@raspberrypi:~ $ hcitool scan
Scanning ...
        08:60:6E:A5:BB:B4      Nexus 7
pi@raspberrypi:~ $
```

The above screenshot shows that a Nexus 7 tablet has been detected. (Take note of the hardware device address of your N7.) If your RPi is not able to detect your N7, turn-off, and then turn-on the Bluetooth on your N7, and check that it is made discoverable (i.e. visible) to nearby devices.



Nexus 7 is visible to nearby devices while Bluetooth settings is open.

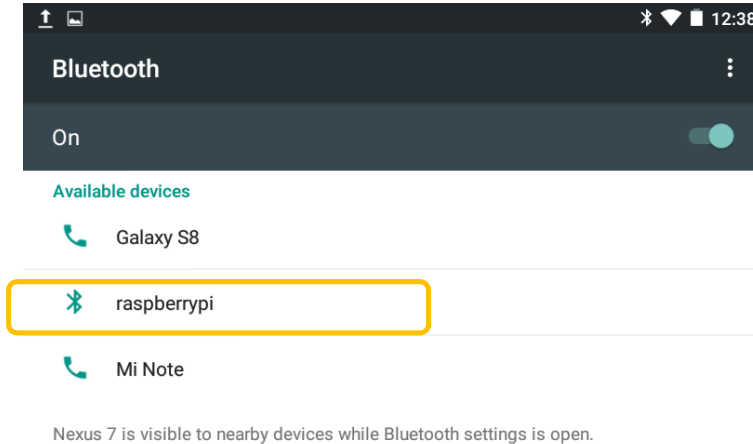
## CE3004/CZ3004 MULTIDISCIPLINARY DESIGN PROJECT

### Notes on Raspberry Pi System Setup for MDP

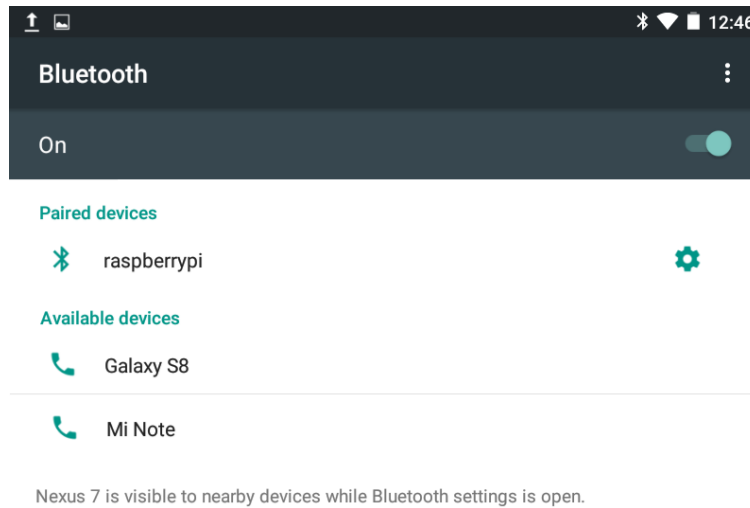
Note that while the RPi can detect the N7, the N7 may not be able to detect the RPi, such as when the RPi is not set to discoverable. To make the RPi also visible to N7, use the command

```
sudo hciconfig hci0 piscan
```

On the N7, refresh (or turn-off and turn-on) the Bluetooth control, which will then detect visible devices nearby, including the RPi.



Press on the 'raspberrypi' to make N7 paired with the RPi.



#### D.4 Bluetooth connection between RPi and N7 using SPP

At this stage, the N7 is ready to communicate with the RPi. You can verify the proper operation of the interface by transferring data between them using the Bluetooth's Serial Port Profile (SPP).

On the N7, run the BT-Term APP that you have installed earlier on, which uses SPP. However, you need to first find out which channel is used by the APP for its bluetooth interface.

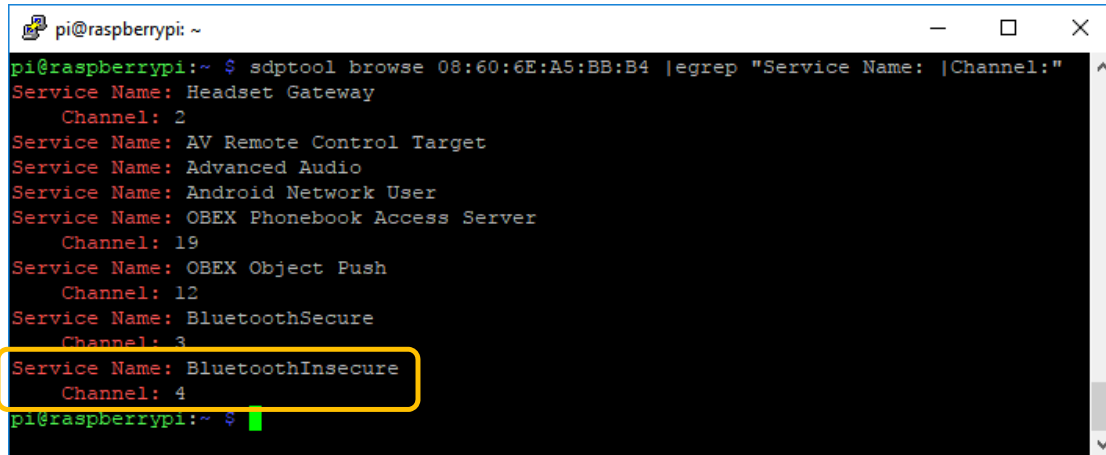


# CE3004/CZ3004 MULTIDISCIPLINARY DESIGN PROJECT

## Notes on Raspberry Pi System Setup for MDP

Back on RPi, run the **sdptool** command as follows to find the SPP service provided by BT-Term APP (use YOUR N7 MAC address):

```
sdptool browse 08:60:6E:A5:BB:B4 | egrep "Service Name: |Channel:"
```

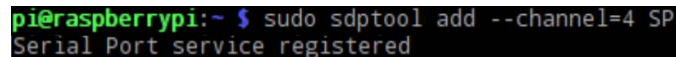


```
pi@raspberrypi:~  
pi@raspberrypi:~ $ sdptool browse 08:60:6E:A5:BB:B4 | egrep "Service Name: |Channel:"  
Service Name: Headset Gateway  
Channel: 2  
Service Name: AV Remote Control Target  
Service Name: Advanced Audio  
Service Name: Android Network User  
Service Name: OBEX Phonebook Access Server  
Channel: 19  
Service Name: OBEX Object Push  
Channel: 12  
Service Name: BluetoothSecure  
Channel: 3  
Service Name: BluetoothInsecure  
Channel: 4  
pi@raspberrypi:~ $
```

This will query the N7 for its available interfaces. Specifically, you are looking for a channel that can support SPP. This is shown to be available through **channel 4**, indicated as “**BluetoothInsecure**” in the screenshot above. (Note: If the BT-Term is not running on N7, this **BluetoothInsecure** Channel will not be detected.)

Add a SP profile for channel number 4

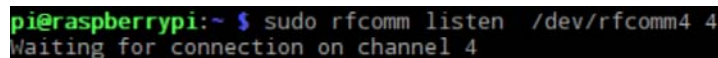
```
sudo sdptool add -channel=4 SP
```



```
pi@raspberrypi:~ $ sudo sdptool add --channel=4 SP  
Serial Port service registered
```

Next, issue the command to place the RPi listening at channel 4, using the **rfcomm listen** command.

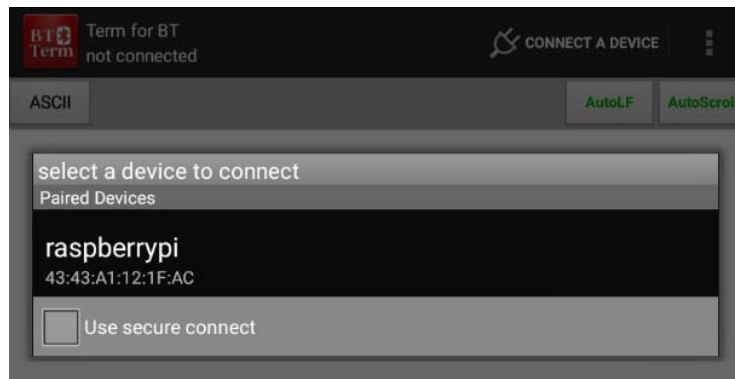
```
sudo rfcomm listen /dev/rfcomm4 4
```



```
pi@raspberrypi:~ $ sudo rfcomm listen /dev/rfcomm4 4  
Waiting for connection on channel 4
```

This puts the RPi in the Slave mode, listening (waiting) for a Master (i.e. N7) to connect through channel 4, which is mapped to a node file **/dev/rfcomm4**

On the N7 tablet, press the **CONNECT A DEVICE** and select **raspberrypi** that was paired earlier on.



# CE3004/CZ3004 MULTIDISCIPLINARY DESIGN PROJECT

## Notes on Raspberry Pi System Setup for MDP

This will make N7 requesting to establish a connection with RPi, and when successful, will display as shown below.



The terminal display in the RPi will also show that the connection is achieved successfully.

```
pi@raspberrypi:~ $ sudo rfcomm listen /dev/rfcomm4 4
Waiting for connection on channel 4
Connection from 08:60:6E:A5:BB:B4 to /dev/rfcomm4
Press CTRL-C for hangup
```

### D.5 Testing

To test the connection, open a new SSH terminal and type

```
echo "Hi from RPi3" > /dev/rfcomm4
```

```
pi@raspberrypi:~ $ echo "Hi from RPi3" > /dev/rfcomm4
pi@raspberrypi:~ $
```

A message should appear on the N7 tablet BT-TERM screen



"Hi from RPi3"

Similarly you can send message from N7 to RPi using its on-screen keyboard, using the **BT-Term**. First open another terminal console. Get the RPi ready to display data in the 'file' /dev/rfcomm4 by typing the following command in the terminal:

```
cat /dev/rfcomm4
```

Then on the N7 BT-Term, type a message (such as "Hello from N7"). The message should also appear on the RPi console display.



```
pi@raspberrypi:~ $ cat /dev/rfcomm4
Hello from N7
```





# CE3004/CZ3004 MULTIDISCIPLINARY DESIGN PROJECT

## Notes on Raspberry Pi System Setup for MDP

Disconnect N7 from the RPi by pressing the on-screen DISCONNECT icon.

Note: It is observed that after you disconnect the Bluetooth connection, you may need to re-start the rfcomm by issuing the command

```
sudo systemctl start rfcomm
```

follows by the

```
sudo rfcomm listen /dev/rfcomm4 4
```

as before.

### D.6 Automate the Bluetooth connection

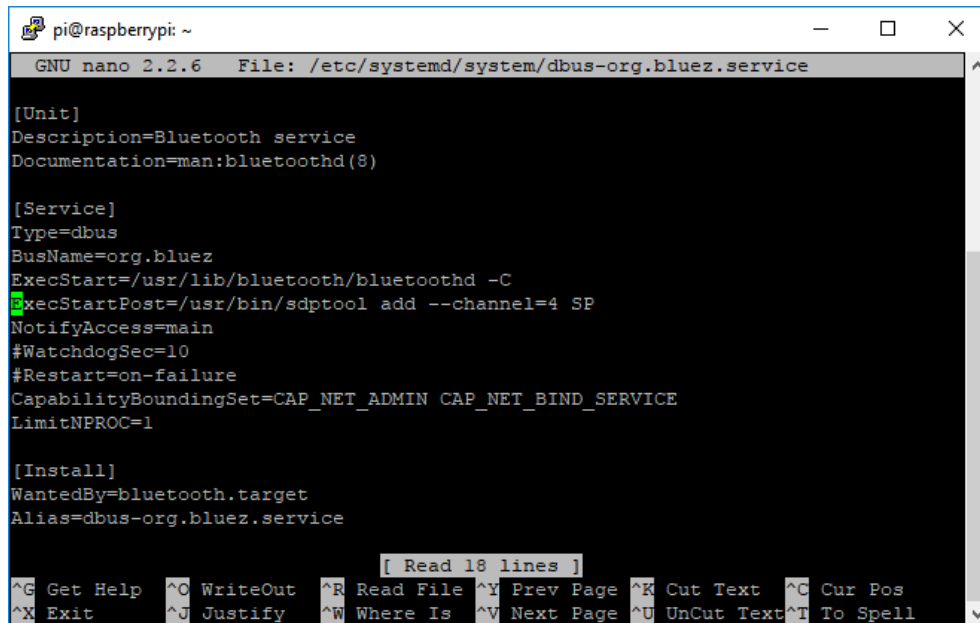
Once you have verified that the Bluetooth connection can operate as expected, you can automate the connection setup process upon boot-up by doing the following.

- (i) Add a new “ExecStartPost=” line immediate after the “ExecStart” line in the file

```
/etc/systemd/system/dbus-org.bluez.service
```

```
ExecStartPost=/usr/bin/sdptool add --channel=4 SP
```

This is to add a SP profile for channel number 4.



```
pi@raspberrypi: ~
GNU nano 2.2.6 File: /etc/systemd/system/dbus-org.bluez.service

[Unit]
Description=Bluetooth service
Documentation=man:bluetoothd(8)

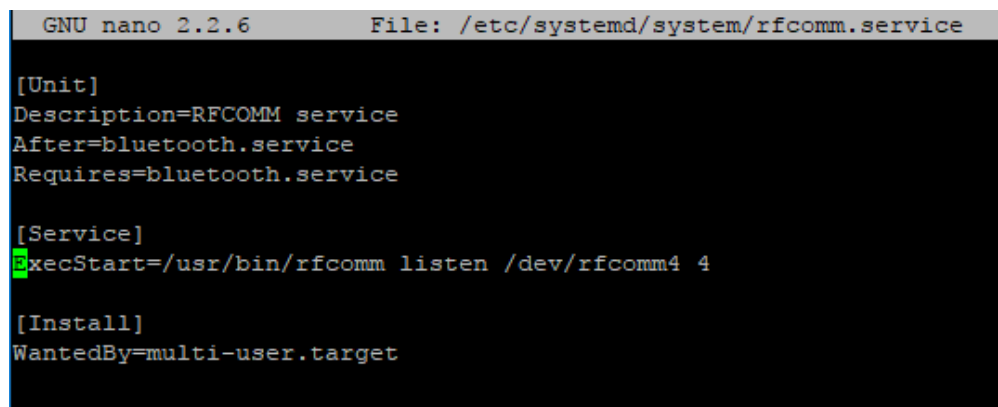
[Service]
Type=dbus
BusName=org.bluez
ExecStart=/usr/lib/bluetooth/bluetoothd -C
ExecStartPost=/usr/bin/sdptool add --channel=4 SP
NotifyAccess=main
#WatchdogSec=10
#Restart=on-failure
CapabilityBoundingSet=CAP_NET_ADMIN CAP_NET_BIND_SERVICE
LimitNPROC=1

[Install]
WantedBy=bluetooth.target
Alias=dbus-org.bluez.service

[ Read 18 lines ]
^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text   ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text ^T To Spell
```

- (ii) Create a file to automate the launching of the rfcomm service upon boot up, using content as shown below

```
sudo nano /etc/systemd/system/rfcomm.service
```



```
GNU nano 2.2.6 File: /etc/systemd/system/rfcomm.service

[Unit]
Description=RFCOMM service
After=bluetooth.service
Requires=bluetooth.service

[Service]
ExecStart=/usr/bin/rfcomm listen /dev/rfcomm4 4

[Install]
WantedBy=multi-user.target
```

# CE3004/CZ3004 MULTIDISCIPLINARY DESIGN PROJECT

## Notes on Raspberry Pi System Setup for MDP

(iii) Restart RPi and upon boot up, you can check whether the SPP service is launched using the command

```
sudo sdptool browse local | egrep "Service Name: |Channel"
```

```
pi@raspberrypi:~$ sudo sdptool browse local | egrep "Service Name: |Channel"
Service Name: Serial Port
Channel: 4
Service Name: Generic Access Profile
Service Name: Generic Attribute Profile
Service Name: AVRCP CT
Service Name: AVRCP TG
pi@raspberrypi:~$
```

(iv) If everything appears OK up to this stage, connect the N7 BT-Term as had been done earlier - and it should work.

### D.7 Monitoring RPi Bluetooth status (Optional)

There is a handy Bluetooth utility that you can use to monitor the connection status of the Bluetooth interface operation in real time.

Open another SSH terminal, and execute the command: **bluetoothctl**

(This utility is part of the **bluez** protocol stack package. If it is not available, it can be installed using:

```
sudo apt-get install bluez
```

)

Once the utility is running, you can start the monitoring by typing the commands:

```
agent on
```

```
scan on
```

```
pi@raspberrypi:~$ bluetoothctl
[NEW] Controller 43:43:A1:12:1F:AC raspberrypi [default]
[NEW] Device 08:60:6E:A5:BB:B4 Nexus 7
[NEW] Device F4:8B:32:E0:C2:1D Mi Note
[bluetooth]# agent on
Agent registered
[bluetooth]# scan on
Discovery started
[CHG] Controller 43:43:A1:12:1F:AC Discovering: yes
[CHG] Device F4:8B:32:E0:C2:1D RSSI: -47
[NEW] Device C0:D3:C0:AD:22:48 Galaxy S8
[NEW] Device CB:7F:77:87:DE:63 Flex 2
[CHG] Device C0:D3:C0:AD:22:48 RSSI: -60
[bluetooth]#
```

If there is any active Bluetooth device detected, it will be displayed on screen, (such as the N7, Mi Note, Galaxy and Flex devices shown above), together with their hardware device addresses.

It will also display the connection status, which is continuously updated in real time.

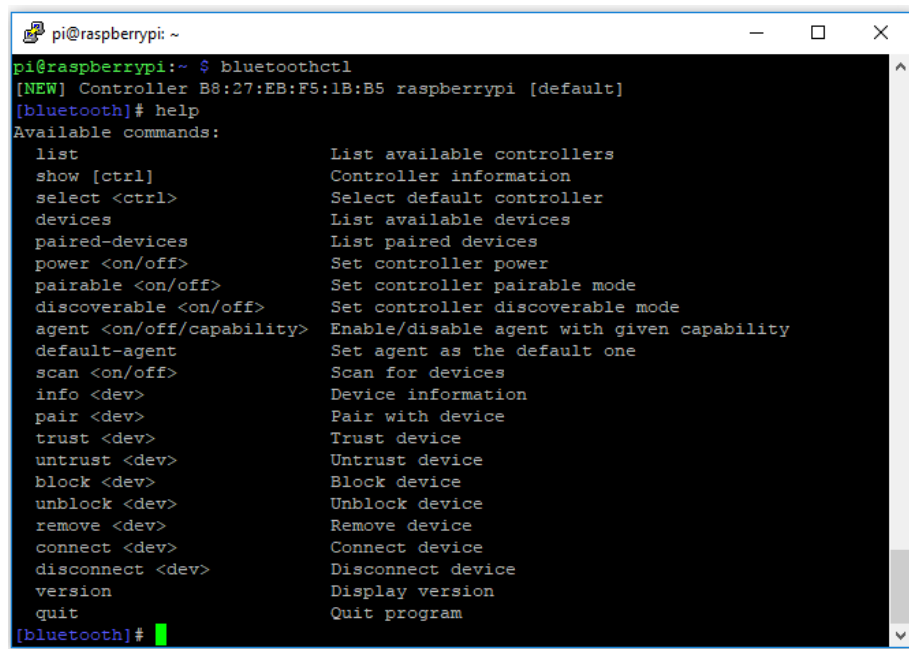
```
pi@raspberrypi:~$ bluetoothctl
[NEW] Controller 43:43:A1:12:1F:AC raspberrypi [default]
[NEW] Device 08:60:6E:A5:BB:B4 Nexus 7
[bluetooth]# scan on
Discovery started
[CHG] Controller 43:43:A1:12:1F:AC Discovering: yes
[CHG] Device 08:60:6E:A5:BB:B4 Connected: yes
[bluetooth]#

pi@raspberrypi:~$ bluetoothctl
[NEW] Controller 43:43:A1:12:1F:AC raspberrypi [default]
[NEW] Device 08:60:6E:A5:BB:B4 Nexus 7
[bluetooth]# scan on
Discovery started
[CHG] Controller 43:43:A1:12:1F:AC Discovering: yes
[CHG] Device 08:60:6E:A5:BB:B4 Connected: yes
[CHG] Device 08:60:6E:A5:BB:B4 Connected: no
[bluetooth]#
```

# CE3004/CZ3004 MULTIDISCIPLINARY DESIGN PROJECT

## Notes on Raspberry Pi System Setup for MDP

You can type 'help' to see the list of commands available in this program.

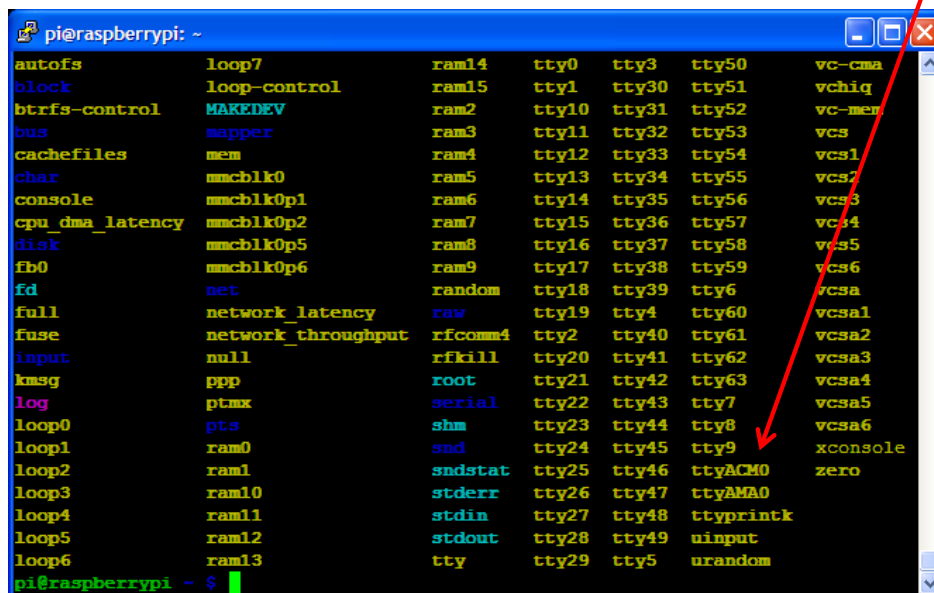


```
pi@raspberrypi:~ $ bluetoothctl
[NEW] Controller B8:27:EB:F5:1B:B5 raspberrypi [default]
[bluetooth]# help
Available commands:
list                List available controllers
show [ctrl]         Controller information
select <ctrl>       Select default controller
devices             List available devices
paired-devices      List paired devices
power <on/off>      Set controller power
pairable <on/off>   Set controller pairable mode
discoverable <on/off> Set controller discoverable mode
agent <on/off/capability> Enable/disable agent with given capability
default-agent       Set agent as the default one
scan <on/off>       Scan for devices
info <dev>          Device information
pair <dev>          Pair with device
trust <dev>         Trust device
untrust <dev>       Untrust device
block <dev>         Block device
unblock <dev>       Unblock device
remove <dev>        Remove device
connect <dev>       Connect device
disconnect <dev>   Disconnect device
version             Display version
quit               Quit program
[bluetooth]#
```

### 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).

As the Arduino set its USB port to appear like a serial port (with default baudrate = 115200) 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:~ $ ls -la /dev
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          vc4
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           rkill          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 hence be sent and displayed through the ttyACM0 file. 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. Installation of OpenCV for Image detection

Starting 2018 Semester 1, a new requirement is added to the MDP: using the RPi to detect image with an on-board camera (NOIR). The recommended software library to implement this is the **OpenCV** (Open Source Computer Vision), which is a library of programming functions aimed at real-time computer vision.

The OpenCV software library the library needs to be compiled from its source code, which can be obtained from the official OpenCV repository. In addition, there are several other libraries (dependencies) that need to be installed before the compilation of the OpenCV source code. Details of how to do these can be found in website such as

<https://www.pyimagesearch.com/2016/04/18/install-guide-raspberry-pi-3-raspbian-jessie-opencv-3/>

It is noted that successful compilation requires quite a large amount of memory. As such it is recommended that the SWAP size of the RPi is increase from 100 to 1024 to provide enough memory for compilation. The step to do this is as follows.

- First type the command in the terminal **free -m** to verify the amount of Swap size

```

pi@raspberrypi: ~
pi@raspberrypi:~ $ free -m
              total        used        free      shared    buffers     cached
Mem:           923          227          695           7         18         128
-/+ buffers/cache:           80          842
Swap:           99           0           99
pi@raspberrypi:~ $

```

- Then edit the file: **sudo nano /etc/dphys-swapfile**
- Change **CONF\_SWAPSIZE=100** to **CONF\_SWAPSIZE=1024**
- Then Restart the dphys-swapfile (which will take a while to complete)

```

# you most likely don't
CONF_SWAPSIZE=1024

```

**sudo /etc/init.d/dphys-swapfile restart**

- Check the setting using **free -m**

```

pi@raspberrypi: ~
pi@raspberrypi:~ $ free -m
              total        used        free      shared    buffers     cached
Mem:           923          228          694           7         18         129
-/+ buffers/cache:           80          842
Swap:           99           0           99
pi@raspberrypi:~ $ sudo /etc/init.d/dphys-swapfile restart
[....] Restarting dphys-swapfile (via systemctl): dphys-swapfile.service
. ok
pi@raspberrypi:~ $
pi@raspberrypi:~ $ free -m
              total        used        free      shared    buffers     cached
Mem:           923          888           34           7         17         776
-/+ buffers/cache:           93          829
Swap:          1023           0          1023
pi@raspberrypi:~ $

```

**Note:** It is important to **set the Swap size back to 100 AFTER** the OpenCV is successfully installed, which otherwise may wear out the SD Card during the operation of the RPi software.

### G. 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
```

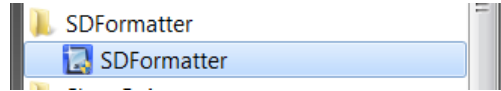
(see [http://elinux.org/R-Pi\\_NAS](http://elinux.org/R-Pi_NAS) for further configuration detail)

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.

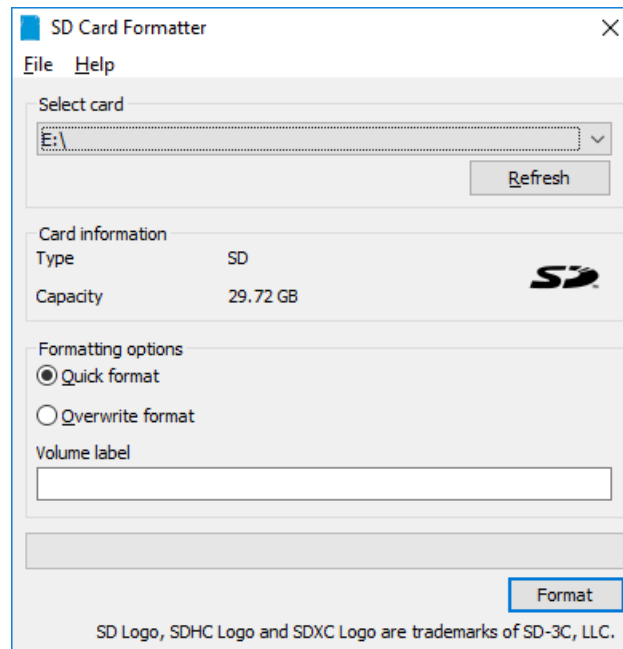


## Appendix A: Installing a new Raspbian to the MicroSD card

- (i) To begin, the MicroSD 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.

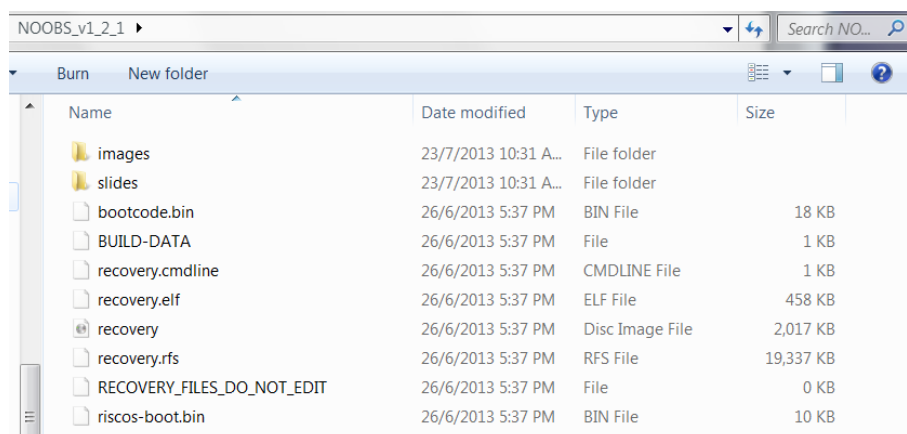


Insert the SD card into the PC. Run the SDFormatter. Select the SD card and format it.



- (ii) Download the latest NOOBS (New Out Of Box Software) program tool from the Raspbian website (at <https://www.raspberrypi.org/downloads/>).

Copy it to your group subdirectory, unzip it and copy all the extracted files (and its subdirectory) to the formatted SD Card.





# CE3004/CZ3004 MULTIDISCIPLINARY DESIGN PROJECT

## Notes on Raspberry Pi System Setup for MDP

**Note:** Alternatively, you may want to consider installing the Raspbian image file (.zip or .img) directly (i.e. instead of using the NOOBS tool), especially if you want to use an earlier version of the Raspbian that is known to be stable, or contain certain required features (e.g. kernel header files for modules development).

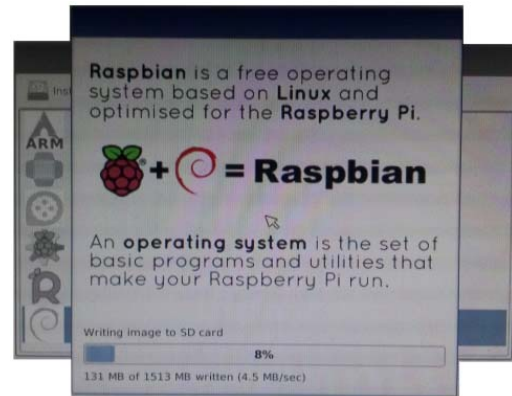
Various Raspbian images are available at <http://downloads.raspberrypi.org/raspbian/images/> for download. To install an image file, you will need to use an image writing tool such as **Etcher** (<https://etcher.io/>) to install the image you have downloaded onto your SD card.

- (iii) Insert the Micro SD 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 (i.e. if you are using the NOOBS). For MDP, choose Raspbian. The RPi will then start the installation process.
- (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 you 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**)

## Appendix B: Script file to test Bluetooth connection (with Nexus 7)

Create the following script file on RPi: `sudo nano bluetooth-test.py`

```
#!/usr/bin
from bluetooth import *
server_sock = BluetoothSocket(RFCOMM)
server_sock.bind(("",PORT_ANY))
server_sock.listen(1)
port = server_sock.getsockname()[1]
uuid = "94f39d29-7d6d-437d-973b-fba39e49d4ee"
advertise_service( server_sock, "MDP-Server",
                    service_id = uuid,
                    service_classes = [ uuid, SERIAL_PORT_CLASS ],
                    profiles = [ SERIAL_PORT_PROFILE ],
#
                    protocols = [ OBEX_UUID ]
                    )

print("Waiting for connection on RFCOMM channel %d" % port)

client_sock, client_info = server_sock.accept()
print("Accepted connection from ", client_info)

try:
    while True:
        print ("In while loop...")
        data = client_sock.recv(1024)
        if len(data) == 0: break
        print("Received [%s]" % data)
        client_sock.send(data + " i am pi!")
except IOError:
    pass

print("disconnected")

client_sock.close()
server_sock.close()
print("all done")
```