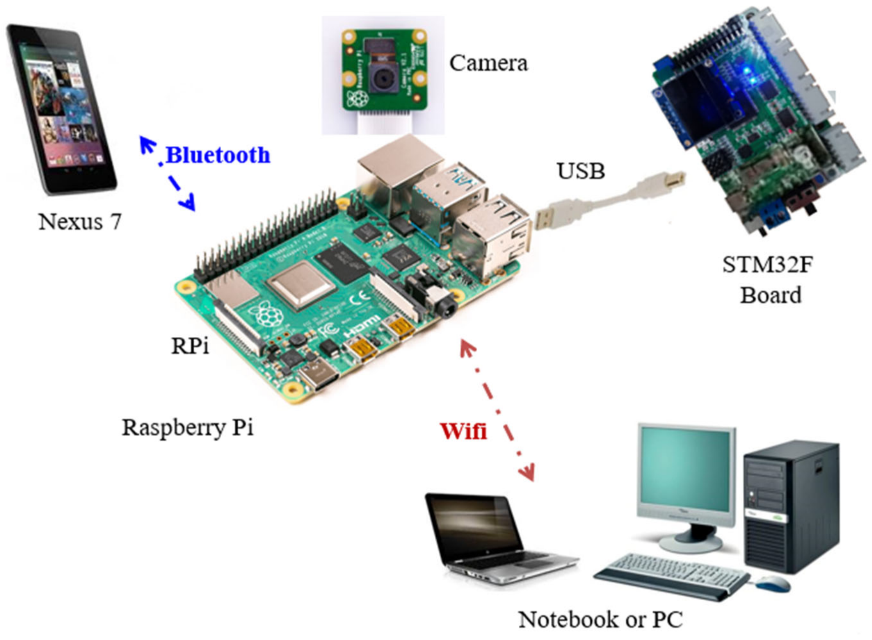
CE3004/CZ3004 MULTIDISCIPLINARY DESIGN PROJECT

Notes on Raspberry Pi System Setup for MDP



System Overview

The Raspberry Pi (RPi 4) runs the Raspberry Pi operating system[1] 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 may be connected to the LAN network point in the Lab, to provide a gateway to the Internet for the other computing devices.

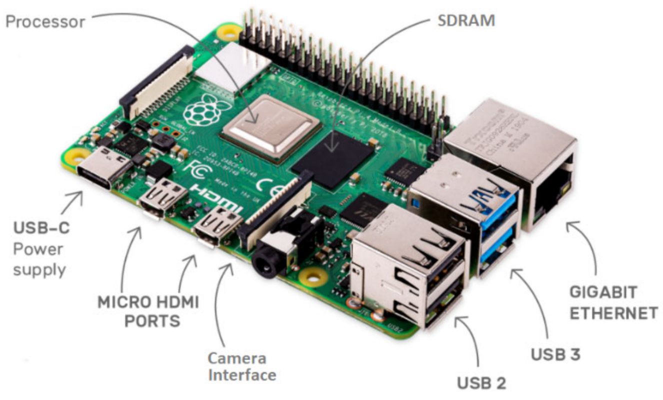
RPi's interface with the Nexus 7 (N7) tablet will be through Bluetooth connection using the rfcomm protocol. RPi interface with the STM32F microcontroller based board will be through a USB connection, using the UART over USM serial interface (ttyUSBx) .

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[2] 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 on the RPi, which allow you to transfer the files over the network and store them on your own PC or thumbdrive.

The figure on the right shows the main components and interfaces on the Raspberry Pi 4 development board provided for this project. It main processor is based on a 64-bit quad-core ARM Cortex-A72 CPU. RPi board was originally developed for teaching/learning computer programming purpose, but has since been widely used in many practical embedded applications.



[1] Raspberry Pi OS (or Raspbian if you are using earlier version)
The RPi 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] 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[4] is up and running, RPi will be assigned a dynamic IP address by the DHCP server used in the SCSE Lab's LAN. However, we need to know the [3]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[3] allocated by the DHCP server during the bootup.

[3] 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 case, you will need the local keyboard and monitor to login and check by using the `ifconfig eth0` command.

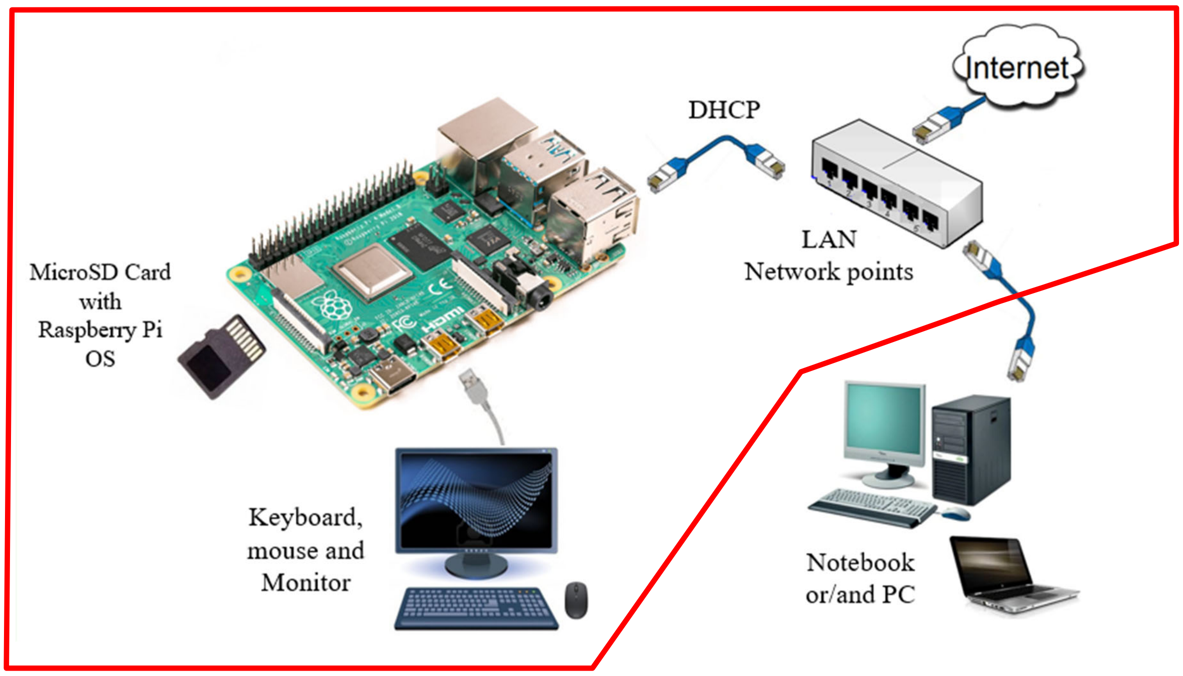
[4] Wireless Access Point and Internet gateway
The most convenient setup is to not connect the local monitor (and keyboard), but be able to SSH RPi through the network. For this, we need a fixed IP address to access RPi. This can be done by setting up the RPi as a Wifi Access Point which uses a fixed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

(Aside: Power Supply in earlier version of RPi
Earlier version of RPi 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:

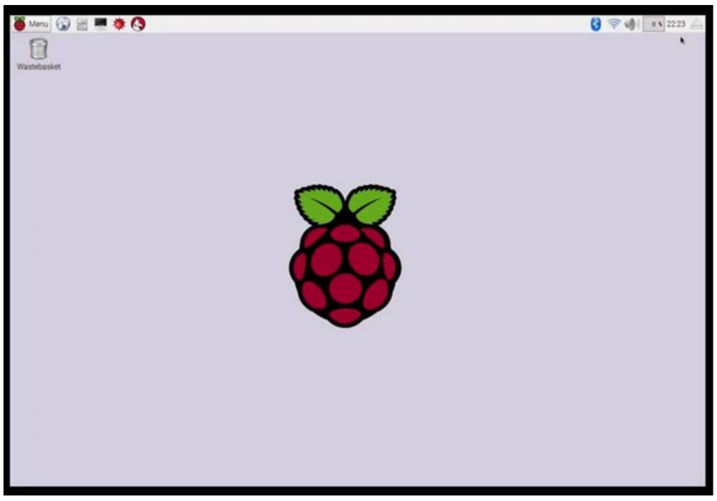
A. Booting up the RPI

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.

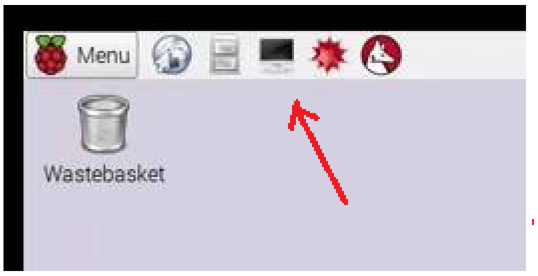

Internet
DHCP
LAN
Network points
MicroSD Card with Raspberry Pi OS
Keyboard, mouse and Monitor
Notebook or/and PC

(i) It is likely that the Micro SD card already contains an OS (e.g. installed by previous batch of MDP students), but it should be re-formatted and install with a new copy of the Raspberry Pi OS - See instruction in Appendix).

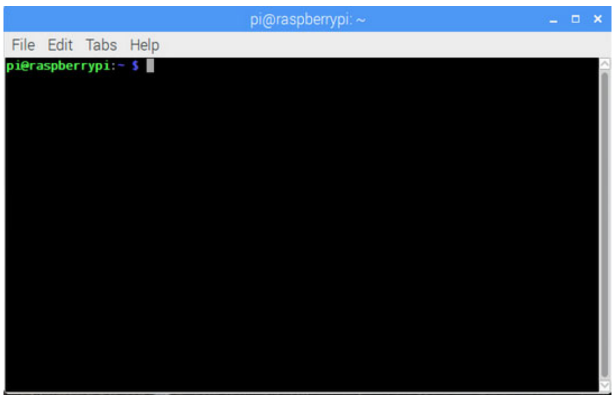
(ii) Plug the Micro SD card into RPi board (beneath the board), apply power to the RPi through the USB-C connector. The RPi should boot up and displays series messages on the local monitor. Eventually it will boot into a GUI screen similar to as shown below.



(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 – also known as the command line **console**, waiting for user to enter the necessary command.

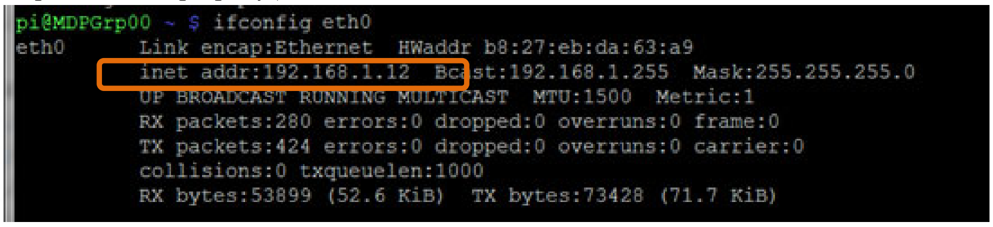


```
pi@raspberrypi:~ $
```

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: **`hostname -I`** or **`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[3].

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

```
ping 8.8.8.8
```

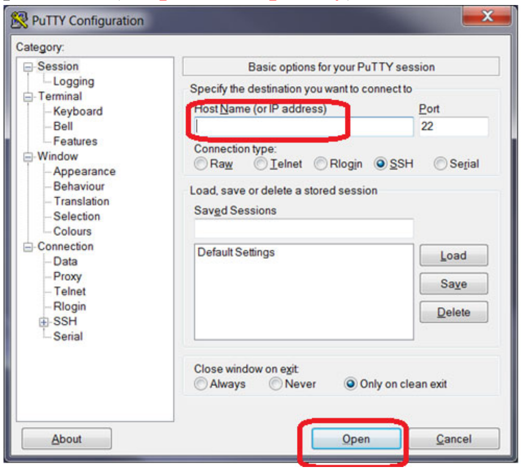
(Google to check who holds this auspicious IP address, which is a DNS computer on the internet)

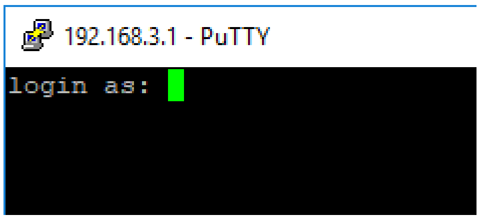
Note: in newer version of RPi, the Ethernet port Network interface may appear with name formed from a prefix `en` followed by `x` (indicating MAC) and its MAC address e.g. `enxb827eb123456`.

(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.

(It is likely that you need to enable the SSH server on the RPi - latest RPi OS has SSH disabled by default. To do so, on the RPi board, execute the command **`sudo raspi-config`** and select **`Inteface Options`** and select Enable SSH server.)

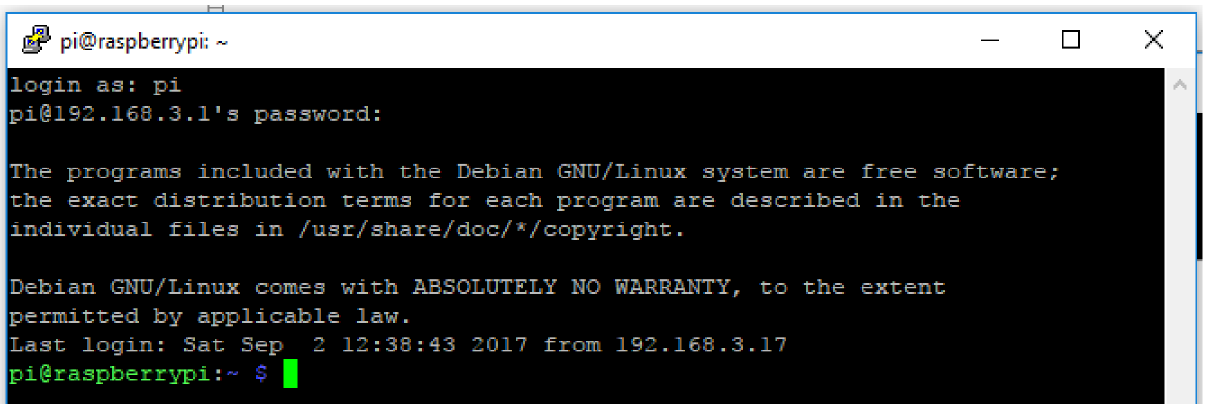
On the networked computer, execute the PuTTY program and configure it with the RPi's IP address. Click Open. A console screen will appear on the remote PC . Log in using the default username and password (i.e. **pi** and **raspberry**).





```
login as:
```

(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)



```
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 12:38:43 2017 from 192.168.3.17
pi@raspberrypi:~ $
```

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[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 setup[5] eventually, as you will achieve through this guide.

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

(iv) Before we proceed further, we will prepare the system for the subsequent configurations. First execute the following commands:

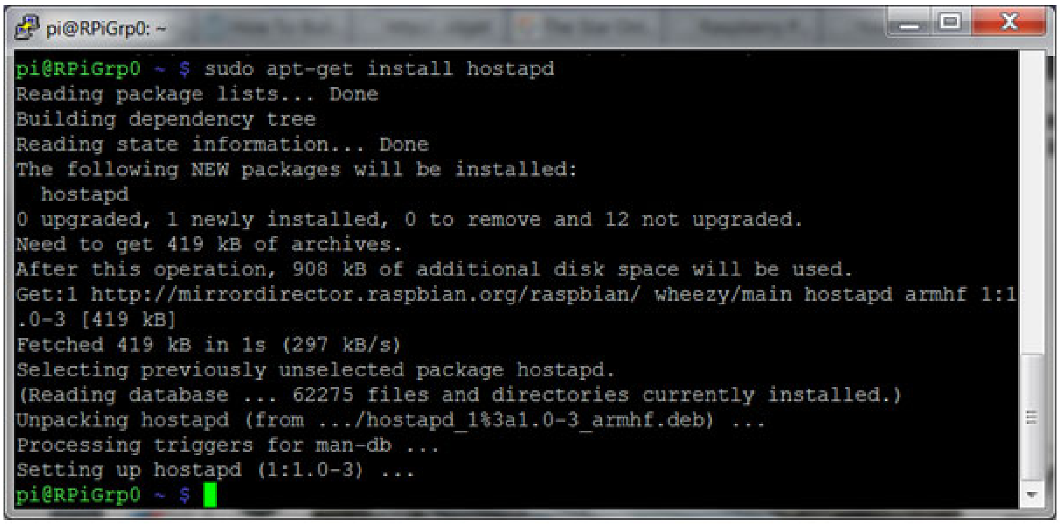
```
sudo apt-get update
sudo apt-get upgrade
```

This is to update the program package lists (from a server on the internet) such that we will always install the most recent version of the program when using the `apt-get` command. (You will need to reboot if you get an upgrade by executing `sudo reboot`)

Next, we will download and install the following 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) The **Host Access Point** package:

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



```
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) The **DNS** and **DHCP** package:

```
sudo apt-get install dnsmasq
```

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, which we will do later.

(c) The **netfilter-persistent** and iptables-persisten packages that are to be used for saving and loading IP rules (the following is one command)

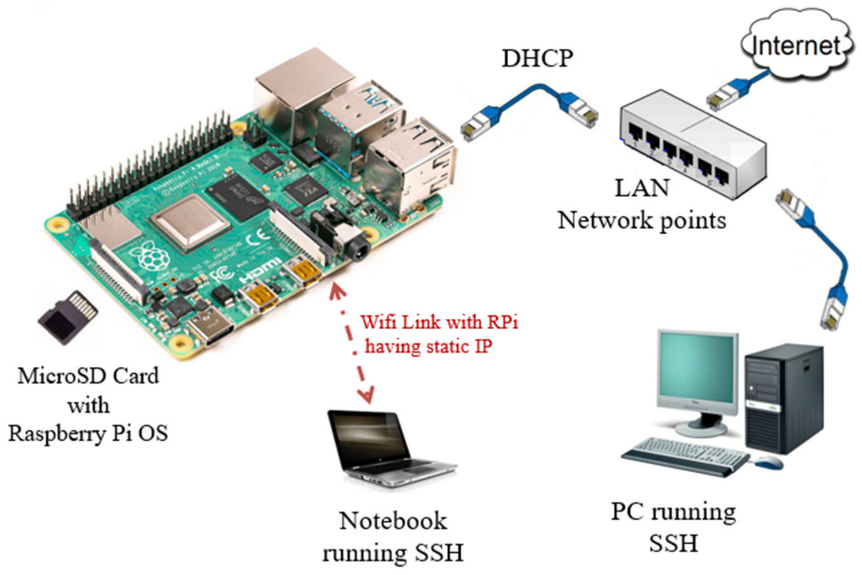
```
sudo DEBIAN_FRONTEND=noninteractive apt install -y \
        netfilter-persistent iptables-persistent
```

Aside: the **noniteractive** is to indicate that we want to install the packages with all the defaults (in this case, '`yes`' as indicated by '`-y`' option) for all questions asked during installing or upgrading the system via apt, i.e. with no interaction.

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, which would not have built-in Wifi, a Wifi dongle (e.g. Edimax) should be plugged into the PC for this exercise. (The 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 has 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 (called **d**aemon) and provide the wireless **host a**ccess **p**oint service (google to find out more detail about this program).

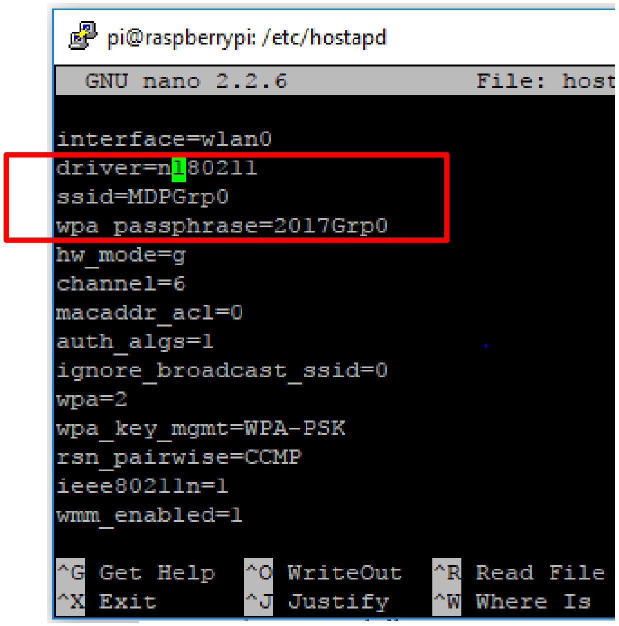
(i) This program can be easily downloaded and installed automatically (with the RPi connected to the internet through eth0) by simply issuing the following Linux command in the SSH client console:

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

(ii) Once the hostapd is installed, 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/`)

(iii) Next create a (new) configuration file `hostapd.conf` 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
```



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

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
```

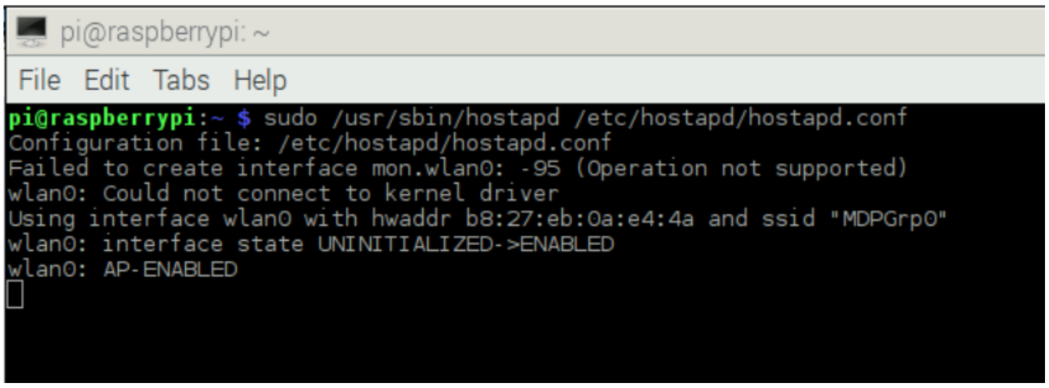
The configuration includes the driver (should be NL80211, letters all in lower case), the ssid of the Wifi hotspot and its password. It is suggested that each group use its group number within the ssid such that the Wifi signal can be clearly identified later.

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.



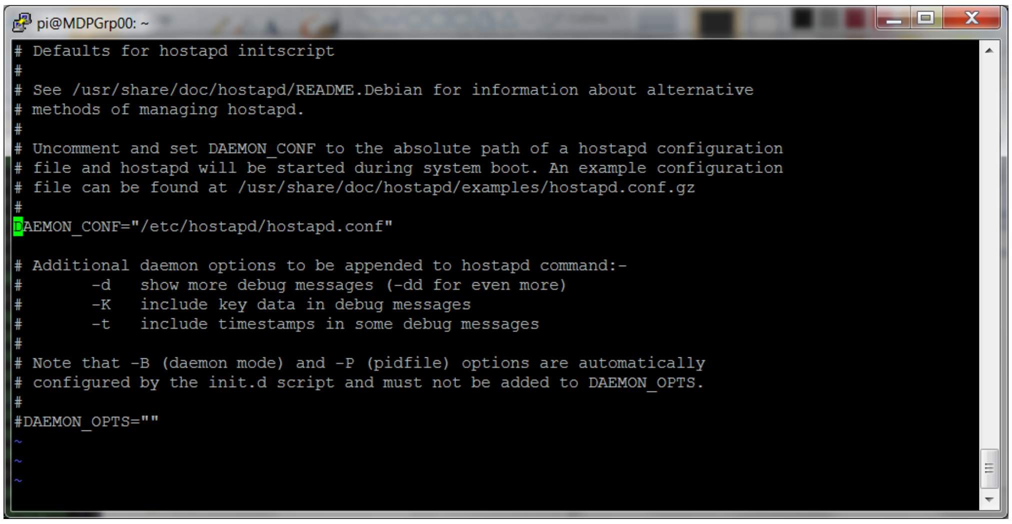
```
pi@raspberrypi:~ $ sudo /usr/sbin/hostapd /etc/hostapd/hostapd.conf
Configuration file: /etc/hostapd/hostapd.conf
Failed to create interface mon.wlan0: -95 (Operation not supported)
wlan0: Could not connect to kernel driver
Using interface wlan0 with hwaddr b8:27:eb:0a:e4:4a and ssid "MDPGrp0"
wlan0: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

(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"**



```
# 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
```

Note: If there is an error message such as

"`Failed to start hostapd.service: Unit hostapd.service is masked`"

Try the following commands:

```
sudo systemctl unmask hostapd
sudo systemctl enable hostapd
sudo systemctl start hostapd
```

C.2 – Setting up the Wireless interface wlan0

Before we configure the DHCP program that we installed earlier, 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). This is done by requesting the DHCP server to assign one as follows:

(i) `sudo nano /etc/dhcpcd.conf`

and add the following lines to the end of the file:

```
interface wlan0
static ip_address=192.168.50.1/24
nohook wpa_supplicant
```

The `/24` is to indicate that the netmask to use is of 24-bit size, which is equivalent to 255.255.255.0.

(Note that recent Linux (and hence RPi OS) by default uses dhcpcd to manage network interfaces, and the file `/etc/network/interfaces` should not be used anymore for network configuration.)

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

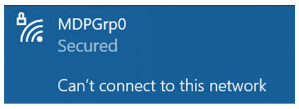
IP addresses 192.168.x.x is reserved for 'private' network, which should not be exposed to the Internet directly (Tunnelling using IP forwarding is used instead to access Internet from private network)

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

(iii) Restart the RPi (e..g using the command **`sudo shutdown -r 0`**), and the network interfaces should then be running and configured with IP address indicated in the **`dhcpcd.conf`** 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 but can't be connected as there is no IP addressed being issued after connection.


MDPGrp0
Secured
Can't connect to this network

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

C.3 – Installing and setting up a DHCP server for the Wifi access point (dnsmasq)

The section is to setup a DHCP server on the RPi. The program that we will use here is the **dnsmasq** (which has been downloaded and installed earlier), but you can use other appropriate one as you prefer – there are available through many references on RPi sites on internet.

(i) With the dnsmasq package installed earlier, there is a new file created: **`/etc/dnsmasq.conf`** which contains a lot of not relevant. So it is easier for us to start from scratch (but save the default configuration file: `sudo mv /etc/dnsmasq.conf /etc/dnsmasq.conf.orig`) by creating a new one.

```
sudo nano /etc/dnsmasq.conf
```

Key in the following two lines in this file (in this example, it is to provide IP addresses between **`192.168.50.11`** and **`192.168.50.11`** for the wlan0 interface.)

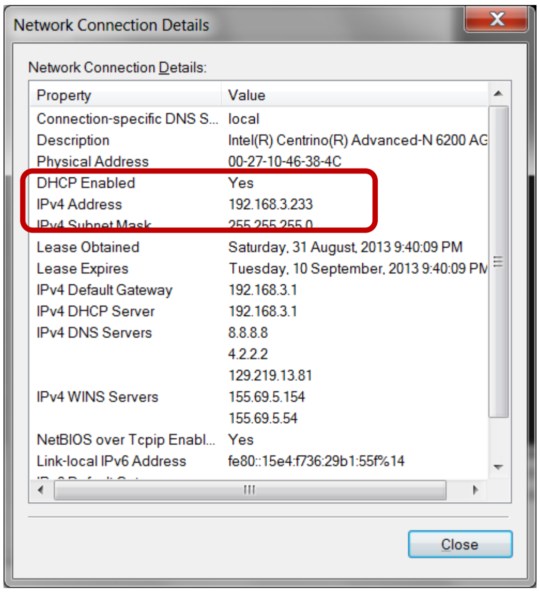
```
interface=wlan0
dhcp-range=192.168.50.11,192.168.50.30,255.255.255.0,24h
```

(Note that the IP addresses setting used here correspond to the 192.138.50.x used in the Wifi's IP example earlier. Each group should hence set the IP addresses accordingly in this file)

(ii) Restart the dnsmasq package

```
sudo systemctl start dnsmasq
```

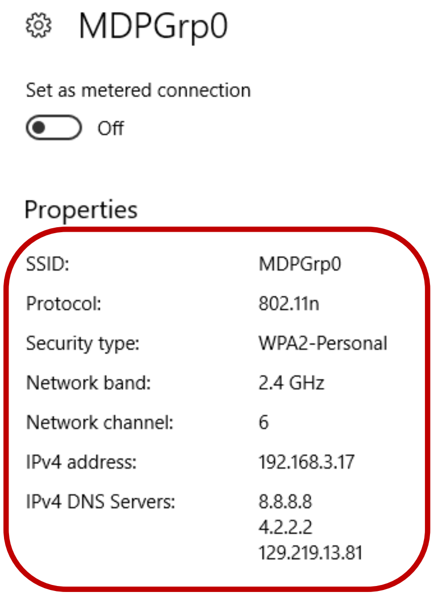
Use the Notebook's Wifi connection to scan and connect to the RPi SSID, and upon successful connection, a dynamically generated IP address will be assigned by the DHCP just set up.



Network Connection Details

Network Connection Details:

Property	Value
Connection-specific DNS S...	local
Description	Intel(R) Centrino(R) Advanced-N 6200 AG
Physical Address	00-27-10-46-38-4C
DHCP Enabled	Yes
IPv4 Address	192.168.3.233
IPv4 Subnet Mask	255.255.255.0
Lease Obtained	Saturday, 31 August, 2013 9:40:09 PM
Lease Expires	Tuesday, 10 September, 2013 9:40:09 PM
IPv4 Default Gateway	192.168.3.1
IPv4 DHCP Server	192.168.3.1
IPv4 DNS Servers	8.8.8.8
	4.2.2.2
	129.219.13.81
IPv4 WINS Servers	155.69.5.154
	155.69.5.54
NetBIOS over Tcpip Enabl...	Yes
Link-local IPv6 Address	fe80::15e4:f736:29b1:55f%14



MDPGrp0

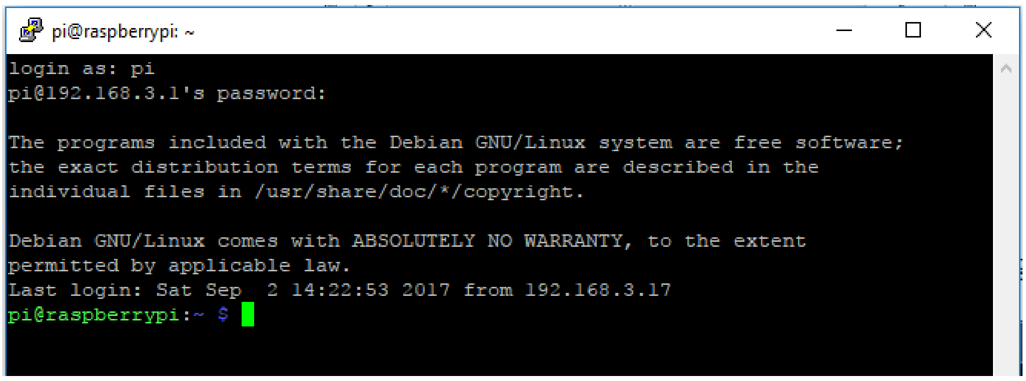
Set as metered connection

Off

Properties

SSID:	MDPGrp0
Protocol:	802.11n
Security type:	WPA2-Personal
Network band:	2.4 GHz
Network channel:	6
IPv4 address:	192.168.3.17
IPv4 DNS Servers:	8.8.8.8 4.2.2.2 129.219.13.81

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 in this example).



```
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:~ $
```

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.

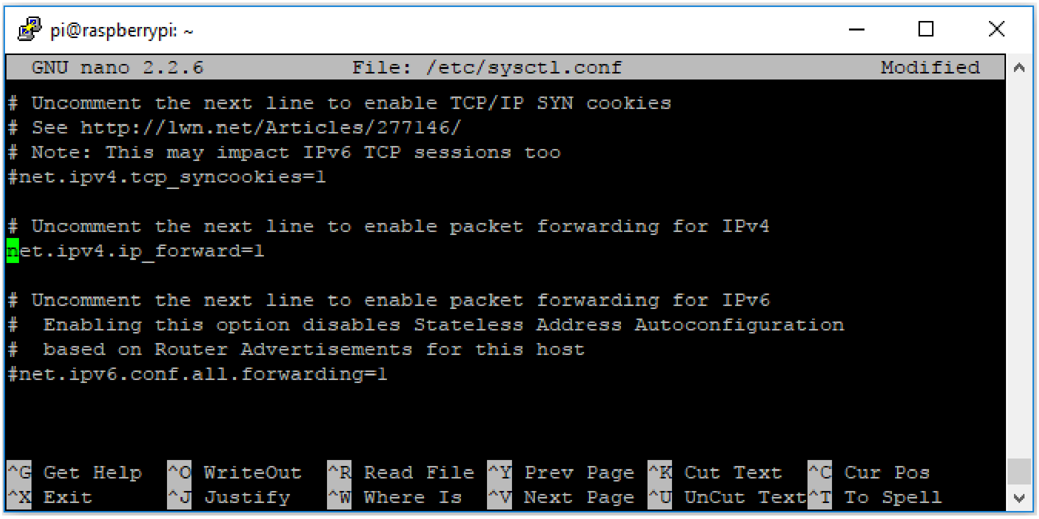
IP forwarding is to be used such that packet received at the Wifi interface wlan0 will be forward to the Ethernet interface eth0 for internet access. (Remember that this is useful during the development work, but will not be needed for the eventually robotic setup since there won't be any wired connection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

(i) To enable IP forwarding in the kernel, we can use the following Linux command:

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

But to make this setup automatically on boot, we can edit the file `/etc/sysctl.conf` and uncomment the following line in the file:

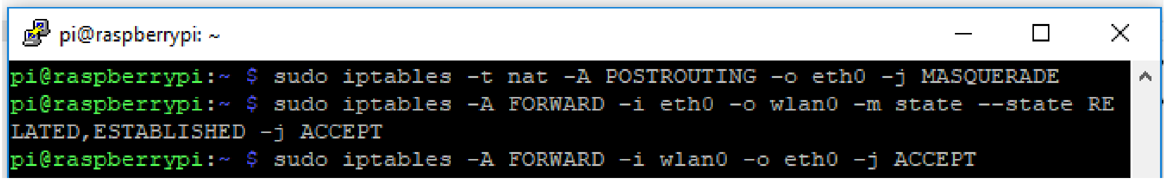
```
net.ipv4.ip_forward=1
```



```
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 we 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
```



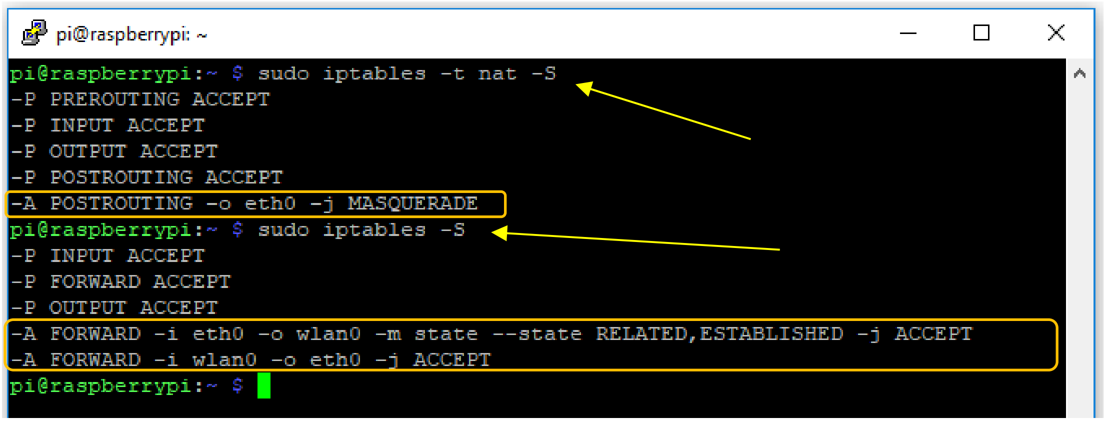
```
pi@raspberrypi: ~
pi@raspberrypi:~ $ sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
pi@raspberrypi:~ $ sudo iptables -A FORWARD -i eth0 -o wlan0 -m state --state RE
LATED,ESTABLISHED -j ACCEPT
pi@raspberrypi:~ $ sudo iptables -A FORWARD -i wlan0 -o eth0 -j ACCEPT
```

We can check the entries in the iptables to confirm the above settings, using commands as shown in the screenshot below.

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

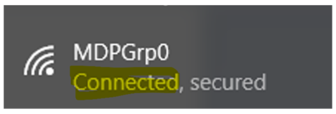
and

```
sudo iptables -S
```



```
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, which can be observed on the notebook Wifi link status, or can be tested by accessing the Internet through a browser on the notebook that is Wifi linked to the RPi board.

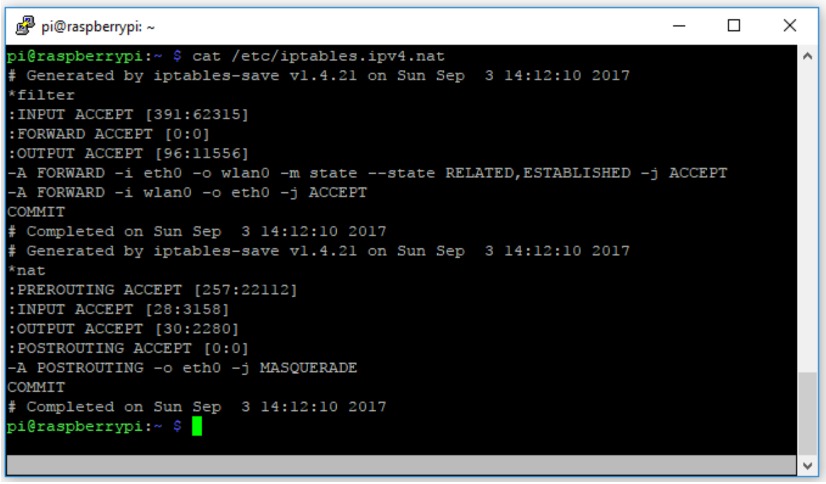


iii) To keep the iptables rules automatically on boot-up, we will save the setting into a file first.

```
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@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:~ $
```

iv) To save these setups such that they will always be loaded upon startup, use the **`netfilter-persistent`** that was installed earlier (see page 6)

```
sudo netfilter-persistent save
```

These will save the filtering rules in the directory **`/etc/iptables`**

v) Now reboot the RPi board to test the setup:

```
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.

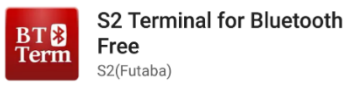
From this point onward, the local monitor and keyboard are really not necessary anymore, as we can always SSH to the RPi board using the Wifi link and then access the internet through the RPi Ethernet connection.

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

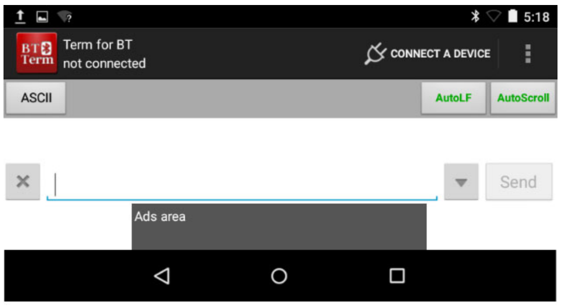
RPi comes with a built-in Bluetooth transceiver which can be used to connect to other Bluetooth device such as the Nexus 7 (N7) tablet. In here, we will try to establish a Bluetooth connection between the RPi and N7 tablet, communicating using the Serial Port profile (SPP).

D.1 Preparation work

(a) To be able to verify the proper configuration of the RPi bluetooth connection, we 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, we will 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.



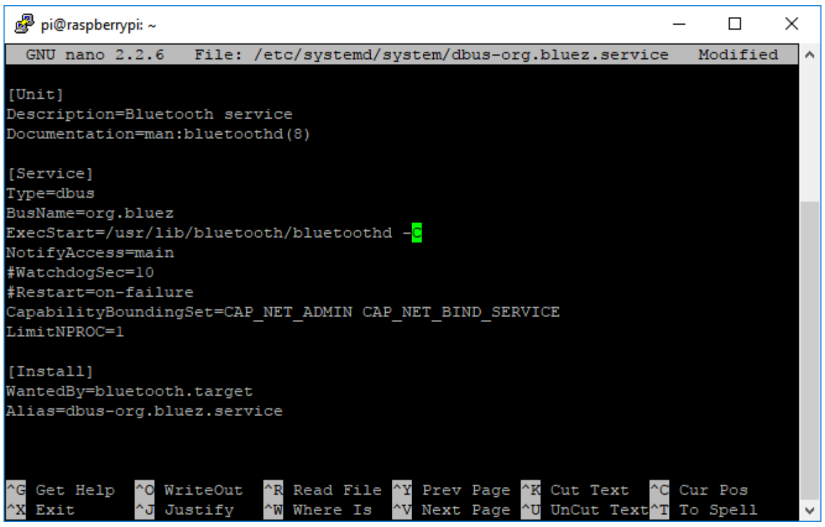
Download and install the BT-Term APP. Test run the APP on the N7 tablet, and it should indicate that it is ready, waiting to connect to a device as shown below. Close the APP as we will need to pair its Bluetooth with RPi first.



(b) For successful interfacing with N7, the RPi bluetooth daemon process need to be run in the ompatible 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.

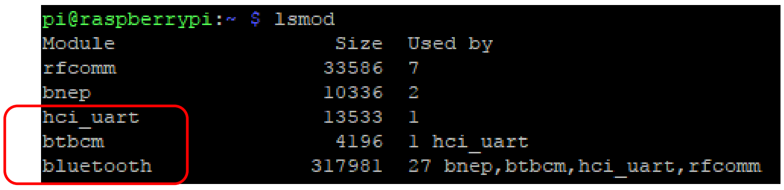


```
pi@raspberrypi: ~
  GNU nano 2.2.6    File: /etc/systemd/system/dbus-org.bluez.service    Modified

[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`.

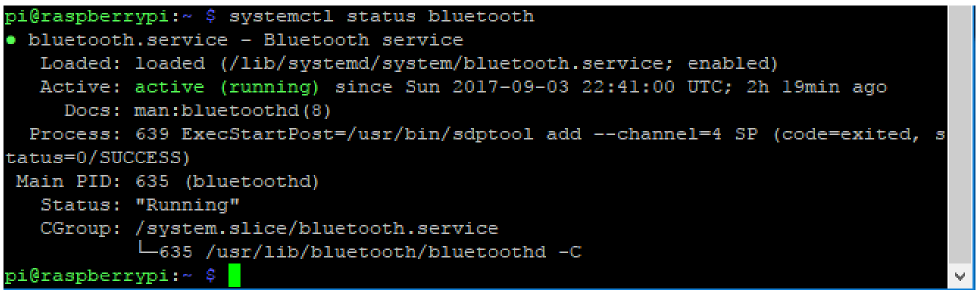
D.2 Checking the RPi Bluetooth status

It is likely that the RPi 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
```

Next check the status of the Bluetooth interface, use the command: **`systemctl status bluetooth`**



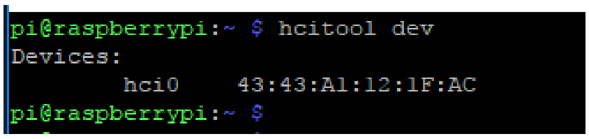
```
pi@raspberrypi:~ $ systemctl status bluetooth
● bluetooth.service - Bluetooth service
   Loaded: loaded (/lib/systemd/system/bluetooth.service; enabled)
   Active: active (running) since Sun 2017-09-03 22:41:00 UTC; 2h 19min ago
     Docs: man:bluetoothd(8)
  Process: 639 ExecStartPost=/usr/bin/sdptool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

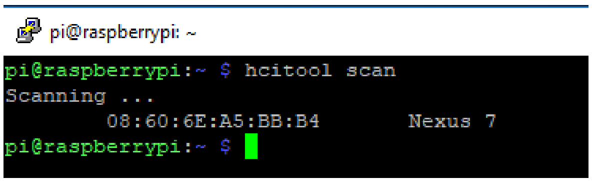
We will be using the **`hcitool`** command to configure the Bluetooth interface.
(Note: At this stage, you might want to also run the **`bluetoothctl`** utility on another terminal that can monitor the bluetooth activities – see D.7)

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



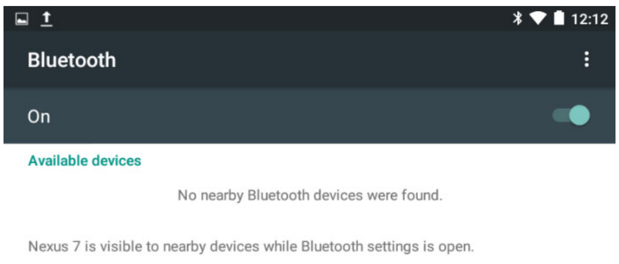
```
pi@raspberrypi:~ $ hcitool dev
Devices:
        hci0    43:43:A1:12:1F:AC
pi@raspberrypi:~ $
```

We can 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.

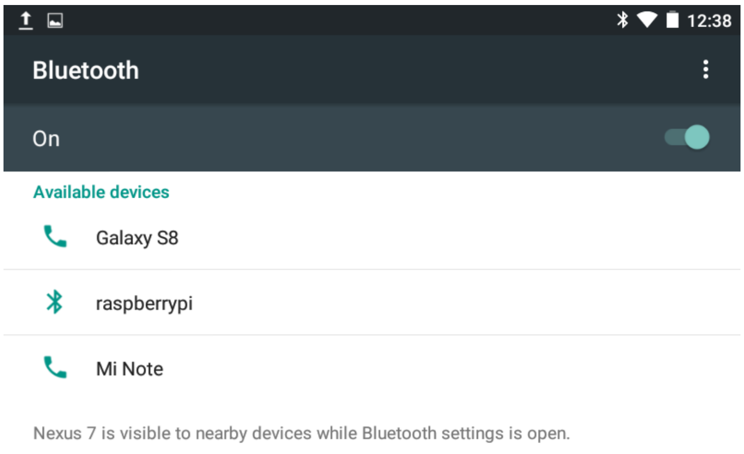


12:12
Bluetooth
On
Available devices
No nearby Bluetooth devices were found.
Nexus 7 is visible to nearby devices while Bluetooth settings is open.

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

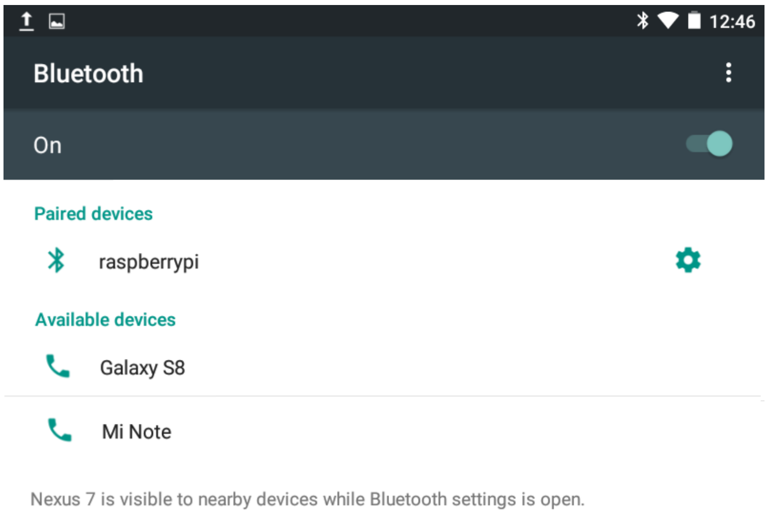
```
sudo hciconfig hci0 piscan
```

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



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

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

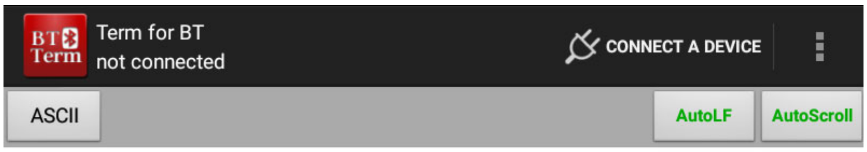


Nexus 7 is visible to nearby devices while Bluetooth settings is open.

D.4 Bluetooth connection between RPi and N7 using SPP

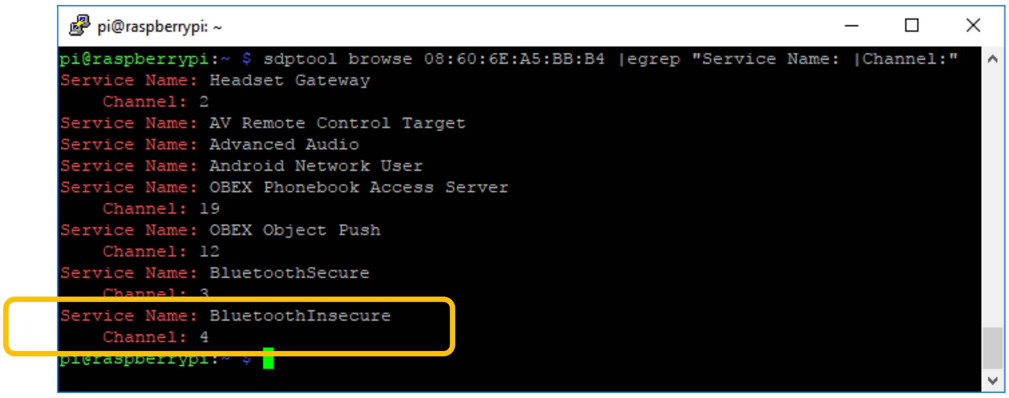
At this stage, the N7 is ready to communicate with the RPi. Next we will verify the proper operation of the interface by transferring data between them using the Bluetooth's Serial Port Profile (SPP).

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



Back on RPi, run the **sdptool** command as follow 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:"
```

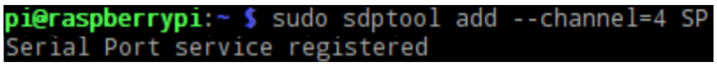


```
pi@raspberrypi:~ $ sdptool browse 08:60:6E:A5:BB:B4 |egrep "Service Name: |Channel:"
Service Name: Headset Gateway
    Channel: 2
Service Name: AV Remote Control Target
Service Name: Advanced Audio
Service Name: Android Network User
Service Name: OBEX Phonebook Access Server
    Channel: 19
Service 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, we 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.)

We hence add a SP profile for channel number 4

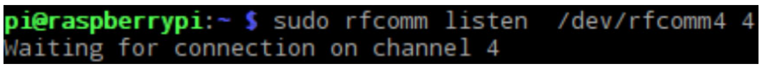
```
sudo sdptool add --channel=4 SP
```



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

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

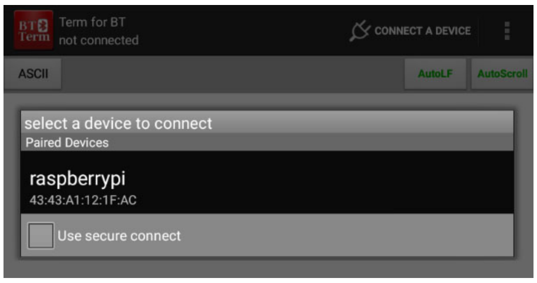
```
sudo rfcomm listen /dev/rfcomm4 4
```



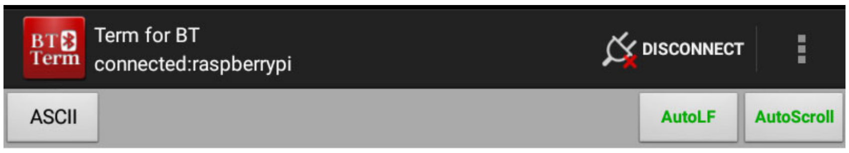
```
pi@raspberrypi:~ $ sudo rfcomm listen  /dev/rfcomm4 4
Waiting for connection on channel 4
```

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

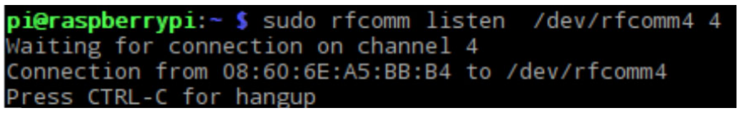
On the N7 tablet, press the CONNECT A DEVICE and select raspberrypi that was paired earlier on.



This will make N7 request to establish a connection with RPi, as shown below



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

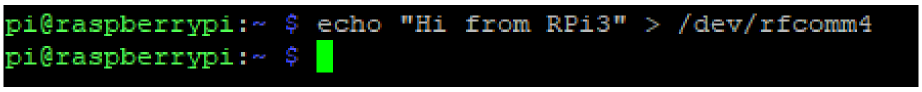


```
pi@raspberrypi:~ $ sudo rfcomm listen  /dev/rfcomm4 4
Waiting for connection on channel 4
Connection from 08:60:6E:A5:BB:B4 to /dev/rfcomm4
Press CTRL-C for hangup
```

D.5 Testing

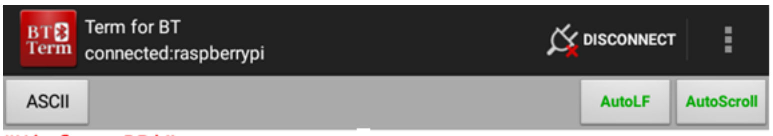
To test the connection, open a new SSH terminal and type

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



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

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

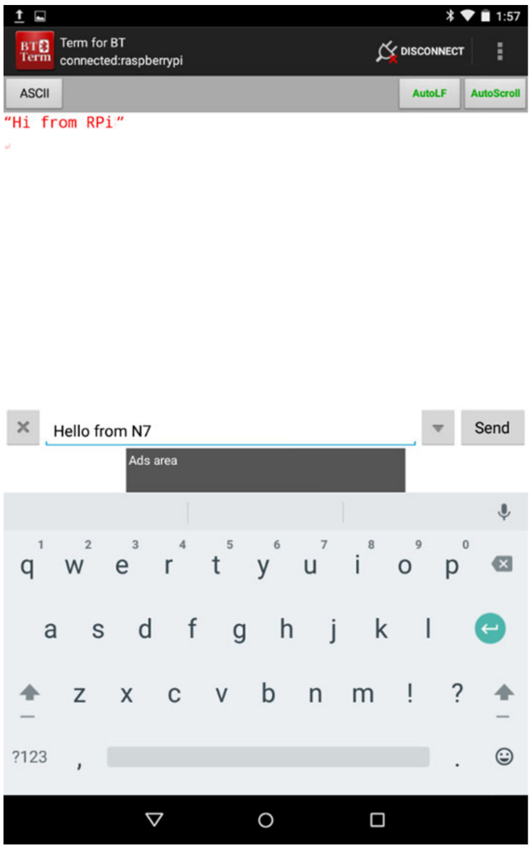


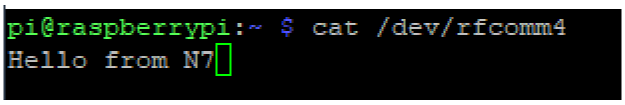
```
"Hi from RPi"
```

Similarly we 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 appear on the RPi console display too.





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

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.

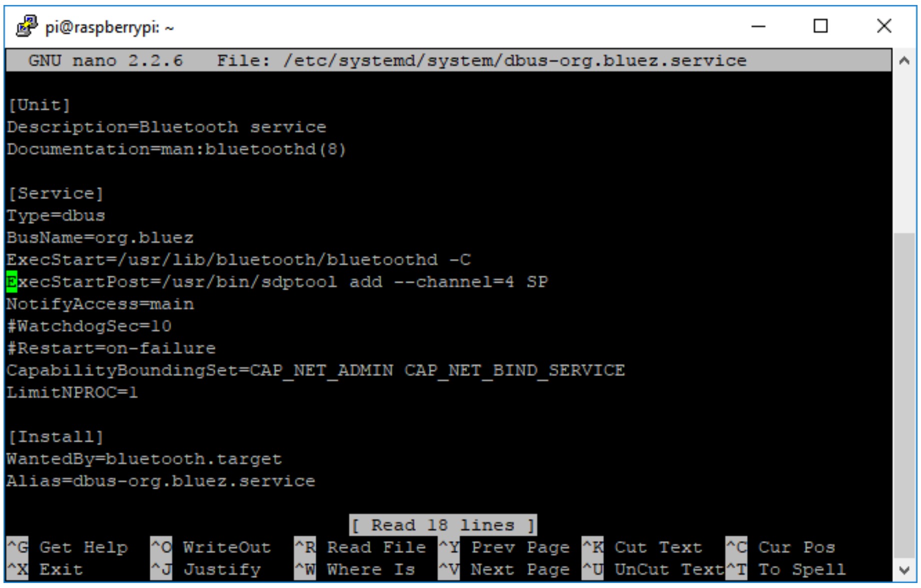
D.6 Automate the Bluetooth connection

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
```

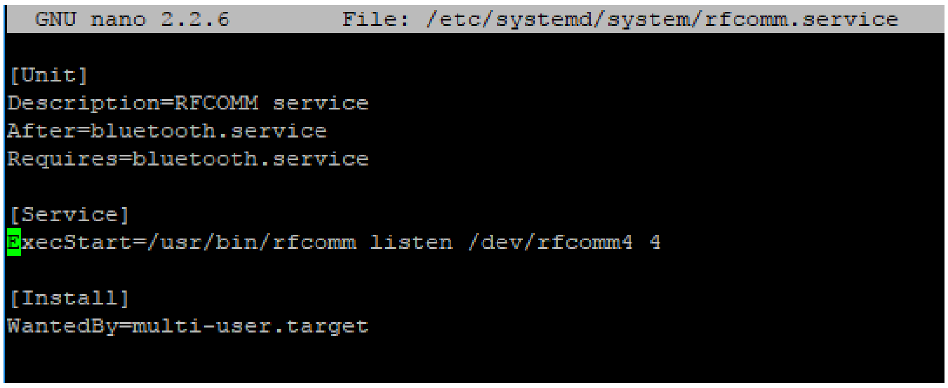
This is to add a SP profile for channel number 4.



```
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

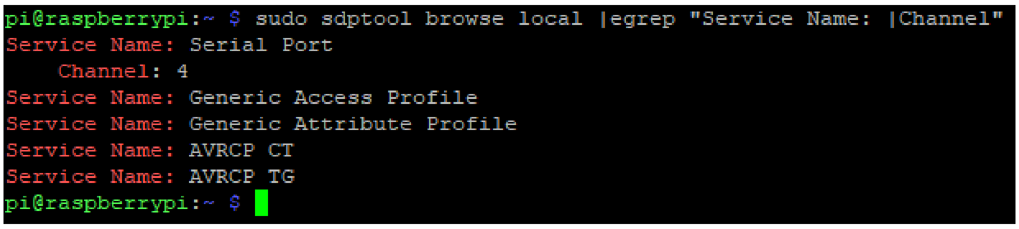
```
sudo nano /etc/systemd/system/rfcomm.service
```



```
  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
```

(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 using bluetoothctl (Optional)

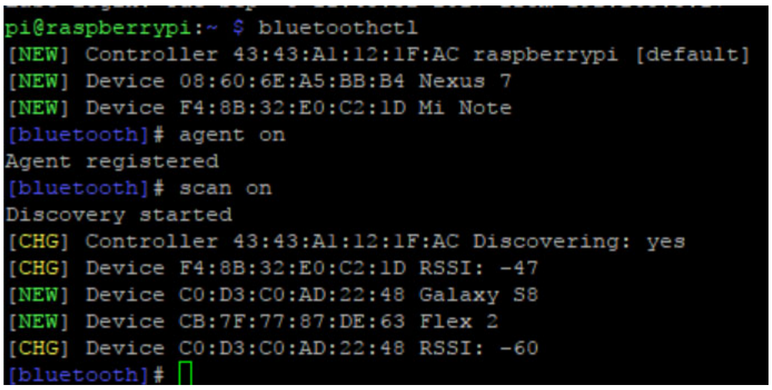
This is a handy Bluetooth utility that you can use to monitor the connection status of the Bluetooth interface operation in real time.

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:

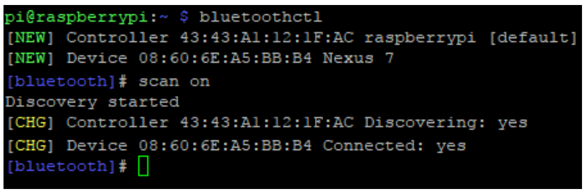
```
agent on
scan on
```



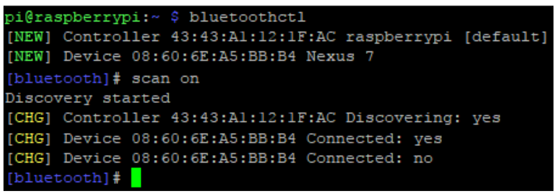
```
pi@raspberrypi:~ $ bluetoothctl
[NEW] Controller 43:43:A1:12:1F:AC raspberrypi [default]
[NEW] Device 08:60:6E:A5:BB:B4 Nexus 7
[NEW] Device F4:8B:32:E0:C2:1D Mi Note
[bluetooth]# agent on
Agent registered
[bluetooth]# scan on
Discovery started
[CHG] Controller 43:43:A1:12:1F:AC Discovering: yes
[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 continuous updated in real time.

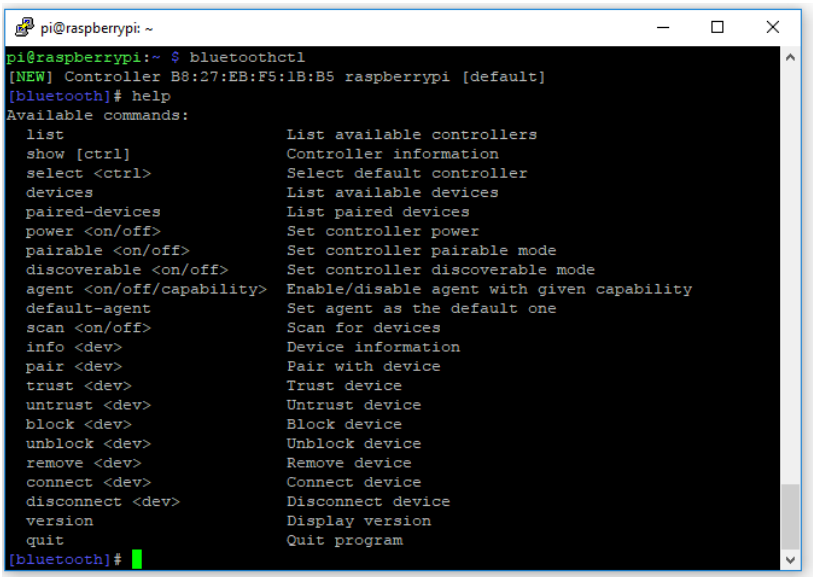


```
pi@raspberrypi:~ $ bluetoothctl
[NEW] Controller 43:43:A1:12:1F:AC raspberrypi [default]
[NEW] Device 08:60:6E:A5:BB:B4 Nexus 7
[bluetooth]# scan on
Discovery started
[CHG] Controller 43:43:A1:12:1F:AC Discovering: yes
[CHG] Device 08:60:6E:A5:BB:B4 Connected: yes
[bluetooth]#
```



```
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]#
```

Type ‘help’ to see the list of commands available in this program.



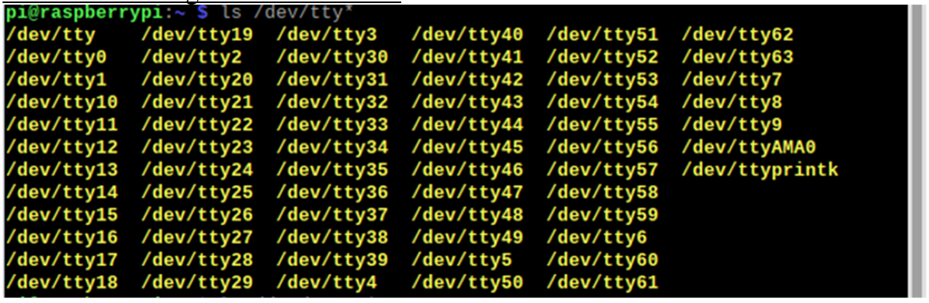
```
pi@raspberrypi:~ $ bluetoothctl
[NEW] Controller B8:27:EB:F5:1B:B5 raspberrypi [default]
[bluetooth]# help
Available commands:
  list                       List available controllers
  show [ctrl]                Controller information
  select <ctrl>              Select default controller
  devices                    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 STM32F Board.

Communication between the RPi and STM32F Board will be through their USB ports. For the STM32F Board board, it is recommended to use its UART2 port that is accessible through the Micro USB port located at the middle of the board (the middle one among the three USB ports).

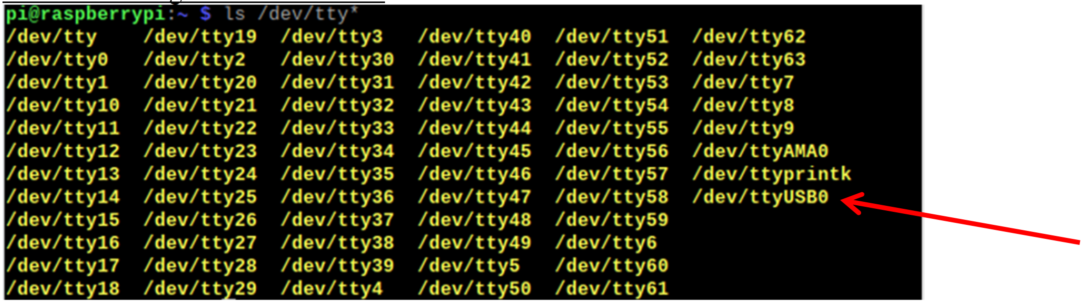
The STM32F Board provides its UART serial interface through its USB port running at the baudrate of 115200. Communication between the two devices can hence be done through the ttyUSBx interface file that is automatically created on connection (typical /dev/ttyUSB0 as shown below).

Before connecting to STM32F Board



```
pi@raspberrypi:~ $ ls /dev/tty*
/dev/tty     /dev/tty19  /dev/tty3   /dev/tty40  /dev/tty51  /dev/tty62
/dev/tty0    /dev/tty2   /dev/tty30  /dev/tty41  /dev/tty52  /dev/tty63
/dev/tty1    /dev/tty20  /dev/tty31  /dev/tty42  /dev/tty53  /dev/tty7
/dev/tty10   /dev/tty21  /dev/tty32  /dev/tty43  /dev/tty54  /dev/tty8
/dev/tty11   /dev/tty22  /dev/tty33  /dev/tty44  /dev/tty55  /dev/tty9
/dev/tty12   /dev/tty23  /dev/tty34  /dev/tty45  /dev/tty56  /dev/ttyAMA0
/dev/tty13   /dev/tty24  /dev/tty35  /dev/tty46  /dev/tty57  /dev/ttyprintk
/dev/tty14   /dev/tty25  /dev/tty36  /dev/tty47  /dev/tty58
/dev/tty15   /dev/tty26  /dev/tty37  /dev/tty48  /dev/tty59
/dev/tty16   /dev/tty27  /dev/tty38  /dev/tty49  /dev/tty6
/dev/tty17   /dev/tty28  /dev/tty39  /dev/tty5   /dev/tty60
/dev/tty18   /dev/tty29  /dev/tty4   /dev/tty50  /dev/tty61
```

After connecting to STM32F Board



```
pi@raspberrypi:~ $ ls /dev/tty*
/dev/tty     /dev/tty19  /dev/tty3   /dev/tty40  /dev/tty51  /dev/tty62
/dev/tty0    /dev/tty2   /dev/tty30  /dev/tty41  /dev/tty52  /dev/tty63
/dev/tty1    /dev/tty20  /dev/tty31  /dev/tty42  /dev/tty53  /dev/tty7
/dev/tty10   /dev/tty21  /dev/tty32  /dev/tty43  /dev/tty54  /dev/tty8
/dev/tty11   /dev/tty22  /dev/tty33  /dev/tty44  /dev/tty55  /dev/tty9
/dev/tty12   /dev/tty23  /dev/tty34  /dev/tty45  /dev/tty56  /dev/ttyAMA0
/dev/tty13   /dev/tty24  /dev/tty35  /dev/tty46  /dev/tty57  /dev/ttyprintk
/dev/tty14   /dev/tty25  /dev/tty36  /dev/tty47  /dev/tty58  /dev/ttyUSB0
/dev/tty15   /dev/tty26  /dev/tty37  /dev/tty48  /dev/tty59
/dev/tty16   /dev/tty27  /dev/tty38  /dev/tty49  /dev/tty6
/dev/tty17   /dev/tty28  /dev/tty39  /dev/tty5   /dev/tty60
/dev/tty18   /dev/tty29  /dev/tty4   /dev/tty50  /dev/tty61
```

Data can hence be sent and displayed through the ttyUSB0 file. To demonstrate the data transfer, the STM32F Board 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
```

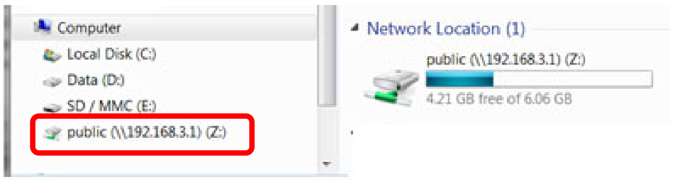
Use a web browser on the connected notebook and key in the RPI's IP address http://192.168.3.1

(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
```

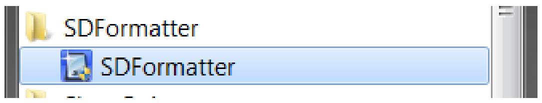
(see http://elinux.org/R-Pi_NAS for further configuration detail)

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.

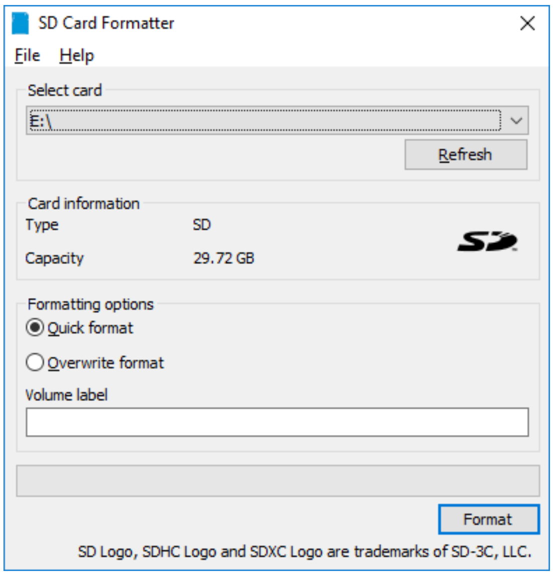


Appendix A: Installing a new Raspbian to the MicroSD card

(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.



SD Card Formatter

File Help

Select card
E:\
Refresh

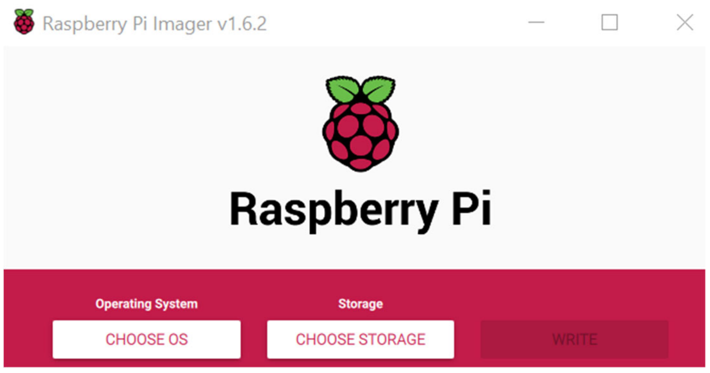
Card information
Type SD
Capacity 29.72 GB

Formatting options
(● Quick format)
○ Overwrite format
Volume label

Format

SD Logo, SDHC Logo and SDXC Logo are trademarks of SD-3C, LLC.

(ii) Go to https://www.raspberrypi.org/software/ to download the Raspberry Pi Imager and install it, following the onscreen instructions.



During the installation, you should select the US Keyboard when prompted.

(iii) After installation is completed, and during the configuration, you may want to select to enable the SSH and the Camera. By default, the Ethernet port is also enabled.

(Note: Configuration can also 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")
```