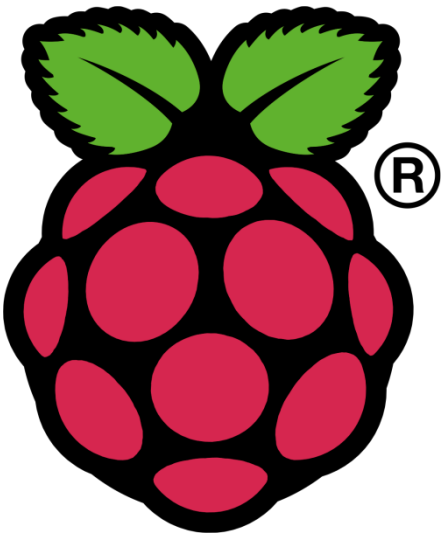
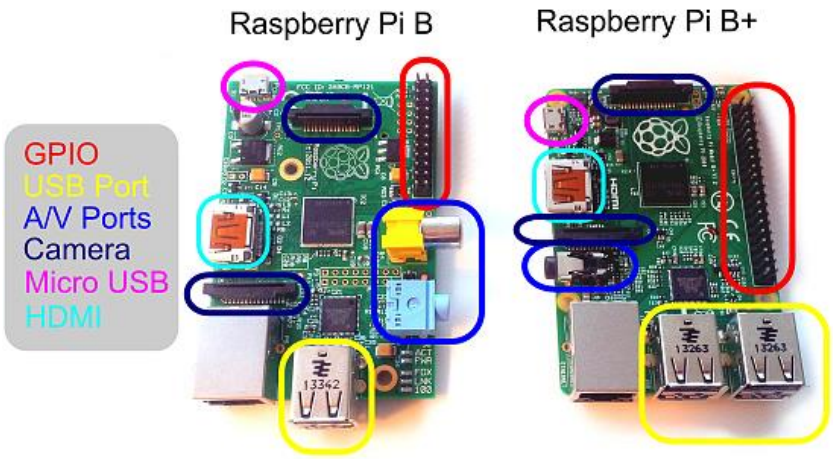


RASPBERRY PI

INSTALLATION GUIDE

REQUIRED ITEMS

- A Raspberry Pi (Either a Model B or Model B+)



- SD Card
  - We recommend an 8GB class 4 SD card.
- Display and connecting cables
  - Any HDMI/DVI monitor or TV should work as a display for the Pi.
  - For best results, use one with HDMI input, but other connections are available for older devices.
- Keyboard and mouse
  - Any standard USB keyboard and mouse will work with your Raspberry Pi.
- Power supply
  - Use a 5V micro USB power supply to power your Raspberry Pi. Be careful that whatever power supply you use outputs at least 5V; insufficient power will cause your Pi to behave unexpectedly.
- Internet connection
  - To update or download software, we recommend that you connect your Raspberry Pi to the internet either via an Ethernet cable or a WiFi adaptor.
- Sound
  - Headphones, earphones or speakers with a 3.5mm jack will work with your Raspberry Pi.

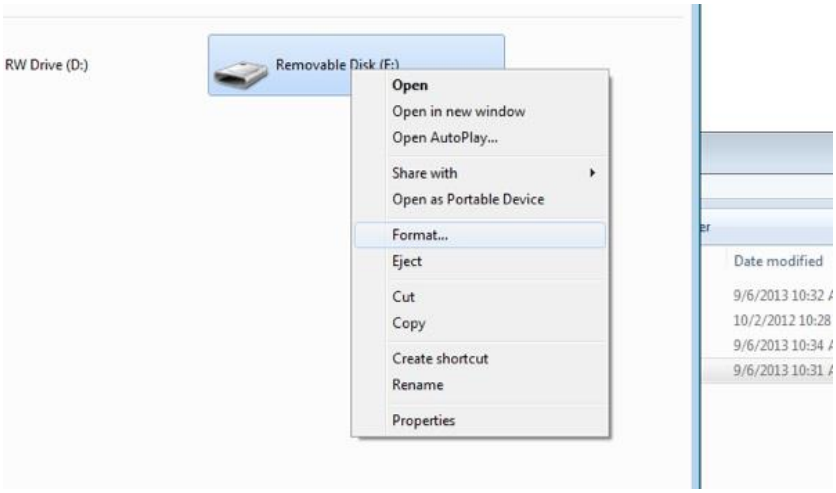
RASPBERRY PI

INSTALLING RASBIAN DEBIAN WHEEZY OPERATING SYSTEM USING WINDOWS

- http://downloads.raspberrypi.org/raspbian_latest
  Dowload the file “RASPBIAN Debian Wheezy.zip” and extract the image file.
- Insert the SD card into your SD card reader(format the sd card) and check which drive letter was assigned. You can easily see the drive letter (for example G:) by looking in the left column of Windows Explorer. You can use the SD Card slot (if you have one) or a cheap SD adaptor in a USB port.
- Download the **Win32DiskImager** utility from the Sourceforge Project page (it is also a zip file); you can run this from a USB drive.
  http://sourceforge.net/projects/win32diskimager/files/latest/download
- Extract the executable from the zip file and run the **Win32DiskImager** utility; you may need to run the utility as administrator. Right-click on the file, and select Run as administrator.
- Select the image file you extracted above.
- Select the drive letter of the SD card in the device box. Be careful to select the correct drive; if you get the wrong one you can destroy your data on the computer's hard disk! If you are using an SD card slot in your computer and can't see the drive in the Win32DiskImager window, try using a cheap SD adaptor in a USB port.
- Click Write and wait for the write to complete.
- Exit the imager and eject the SD card

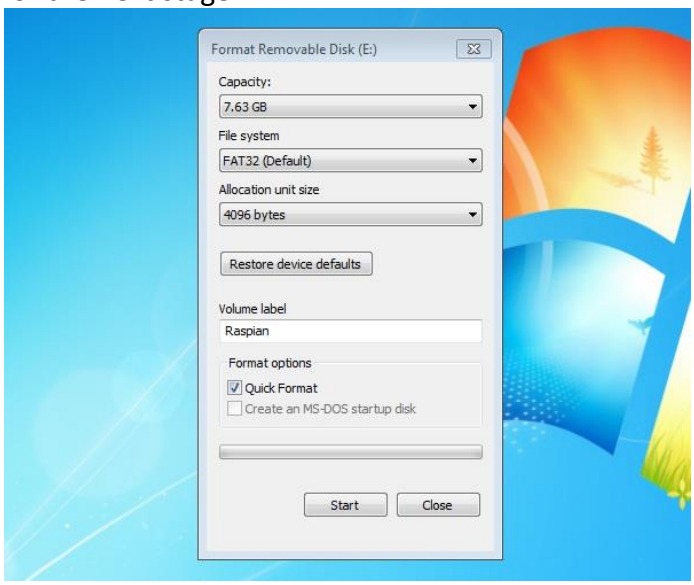
FORMAT THE SD CARD

Locate your SD card drive, in Windows Explorer, and secondary-click the mouse to bring up the context-senstive menu. From the menu select **Format…**. Ensure that the option **FAT32 (Default)** is selected and click **Start**.



Selecting an SD card to format

A few moments later you will see a confirmation that the format has been completed and you SD card is now ready for the next stage.

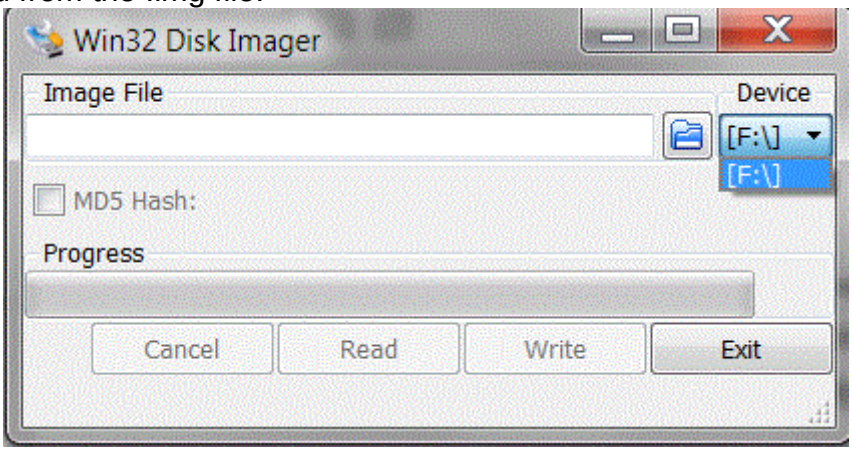


Formatting the SD card

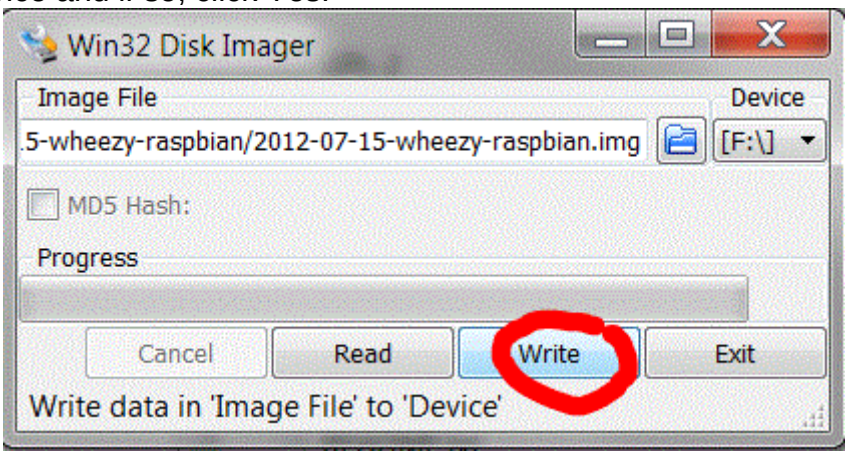
USING WIN32DISKIMAGER

Having plugged in your SD card, (re)start Win32Diskimager. Choose the drive you want to copy the image to (in my case F:).

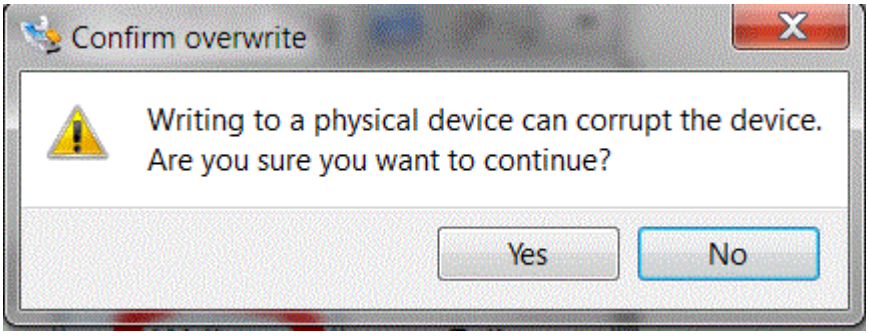
- ***choose the drive with your SD card to write the OS image on***
  Then click on the folder icon and choose the unzipped .img file from earlier that you want to put on the SD card. Then click Write, to write the Operating system on the card from the .img file.



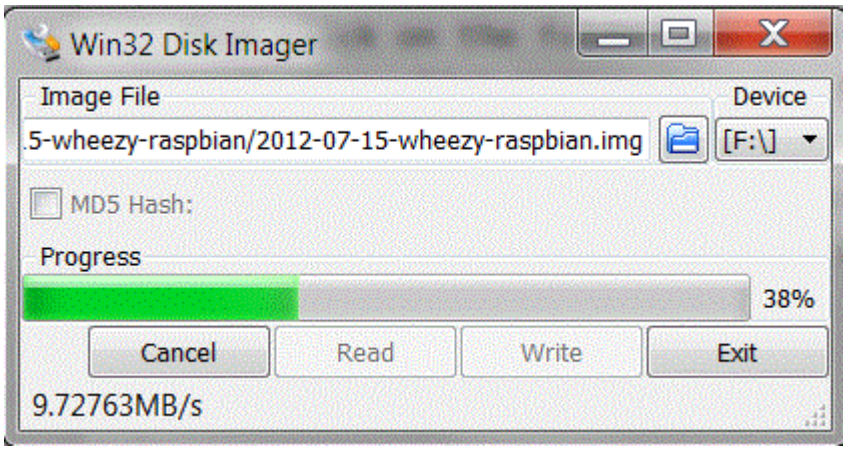
- ***Write OS image from .img file to SD card***
  You will then be asked to confirm. Check carefully that you are writing to the correct device and if so, click Yes.



- ***Check device and confirm***
  The progress bar will show you how far it's got.

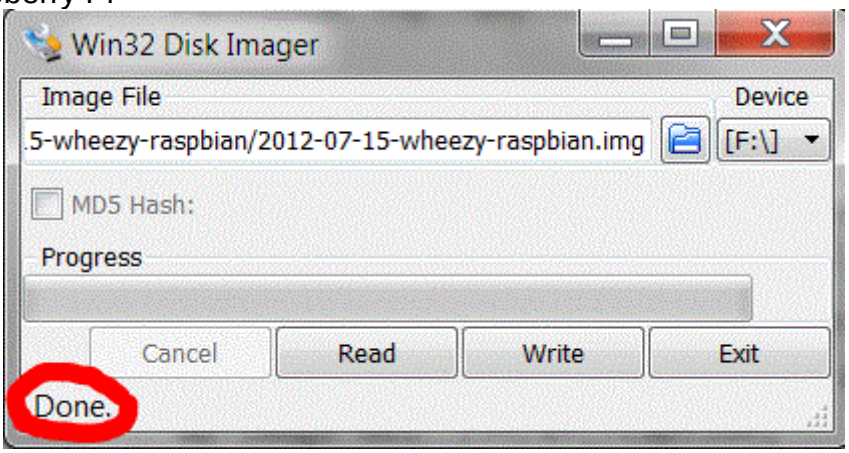


- ***Progress indicator***
  When it's finished it looks like this.



- ***Finished***

Then you can eject the card reader and remove the SD card. Then you can try it out in your Raspberry Pi

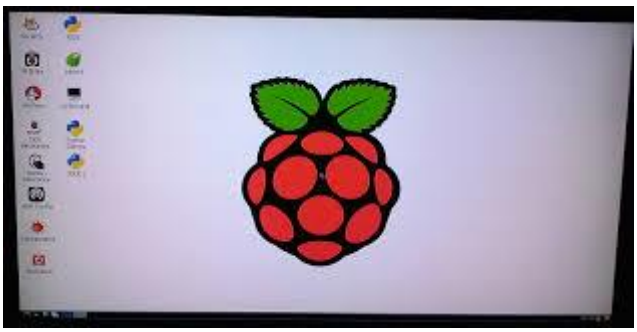


PLUGGING IN YOUR RASPBERRY PI

1. Begin by slotting your SD card into the SD card slot on the Raspberry Pi, which will only fit one way.
2. Next, plug in your USB keyboard and mouse into the USB slots on the Raspberry Pi. Make sure that your monitor or TV is turned on, and that you have selected the right input (e.g. HDMI 1, DVI, etc).
3. Then connect your HDMI cable from your Raspberry Pi to your monitor or TV.
4. If you intend to connect your Raspberry Pi to the internet, plug in an Ethernet cable into the Ethernet port next to the USB ports; if you do not need an internet connection, skip this step.
5. Finally, when you are happy that you have plugged in all the cables and SD card required, plug in the micro USB power supply. This action will turn on and boot your Raspberry Pi.
6. If this is the first time your Raspberry Pi SD card have been used, then you will have to select an operating system and configure it.

LOGGING INTO YOUR RASPBERRY PI

1. Once your Raspberry Pi has completed the boot process, a login prompt will appear. The default login for Raspbian is username pi with the password raspberry. Note you will not see any writing appear when you type the password. This is a security feature in Linux.
2. After you have successfully logged in, you will see the command line prompt **pi@raspberrypi~$.**
3. To load the graphical user interface, type **startx** and press **Enter** on your keyboard.



DOWNLOAD AND INSTALL WIRING PI

WiringPi is maintained under GIT for ease of change tracking, however there is a Plan B if you're unable to use GIT for whatever reasons (usually your firewall will be blocking you, so do check that first!)

ONLINE INSTALL

If you do not have GIT installed, then under any of the Debian releases (e.g. Raspbian), you can install it with:
sudo apt-get install git-core
If you get any errors here, make sure your Pi is up to date with the latest versions of Raspbian:

sudo apt-get update
sudo apt-get upgrade

To obtain WiringPi using GIT:
git clone git://git.drogon.net/wiringPi
If you have already used the clone operation for the first time, then

cd wiringPi
git pull origin
Will fetch an updated version then you can re-run the build script below.

To build/install there is a new simplified script:

cd wiringPi
./build
The new build script will compile and install it all for you – it does use the sudo command at one point, so you may wish to inspect the script before running it.

OFFLINE INSTALL

Click on this URL: (it should open in a new page)
https://git.drogon.net/?p=wiringPi;a=summary

Then look for the link marked snapshot at the right-hand side. You want to click on the top one. This will download a tar.gz file with a name like wiringPi-98bcb20.tar.gz. Note that the numbers and letters after wiringPi (98bcb20 in this case) will probably be different – they're a unique identifier for each release.

You then need to do this to install:

tar xfz wiringPi-98bcb20.tar.gz
cd wiringPi-98bcb20
./build
Note that the actual filename will be different – you will have to check the name and adjust accordingly.

TEST WIRINGPI'S INSTALLATION

run the gpio command to check the installation:

gpio -v
gpio readall
That should give you some confidence that it's working OK.

WiringPi is released under the GNU Lesser Public License version 3.

TESTING SERIAL PORT IN RASPBERRY PI

A great way to test out the serial port is to use the minicom program. If you dont have this installed run

sudo apt-get install minicom

Connect your PC to the Raspberry Pi serial port using an appropriate serial port adapter and wiring, then open Putty or a similar serial terminal program on PC side. Setup a connection using the serial port at 9600 baud.

Now run up minicom on the Raspberry Pi using

minicom -b 9600 -o -D /dev/ttyAMA0

What you type into the minicom terminal screen should appear on the serial PC terminal and vice versa.

GETTING RASPBERRY PI WINDOW IN WINDOWS SYSTEM:

To get raspberry pi window we have to download and install two software's PuTTY and Xming. PuTTY is an SSH and telnet client, developed originally by Simon Tatham for the Windows platform. PuTTY is open source software that is available with source code and is developed and supported by a group of volunteers. **Xming** is an X11 display server for Microsoft Windows operating systems, including Windows XP or later. Following steps illustrates the installing and configuring PuTTy and Xming.

Step1:
Download and install PuTTY using below site.

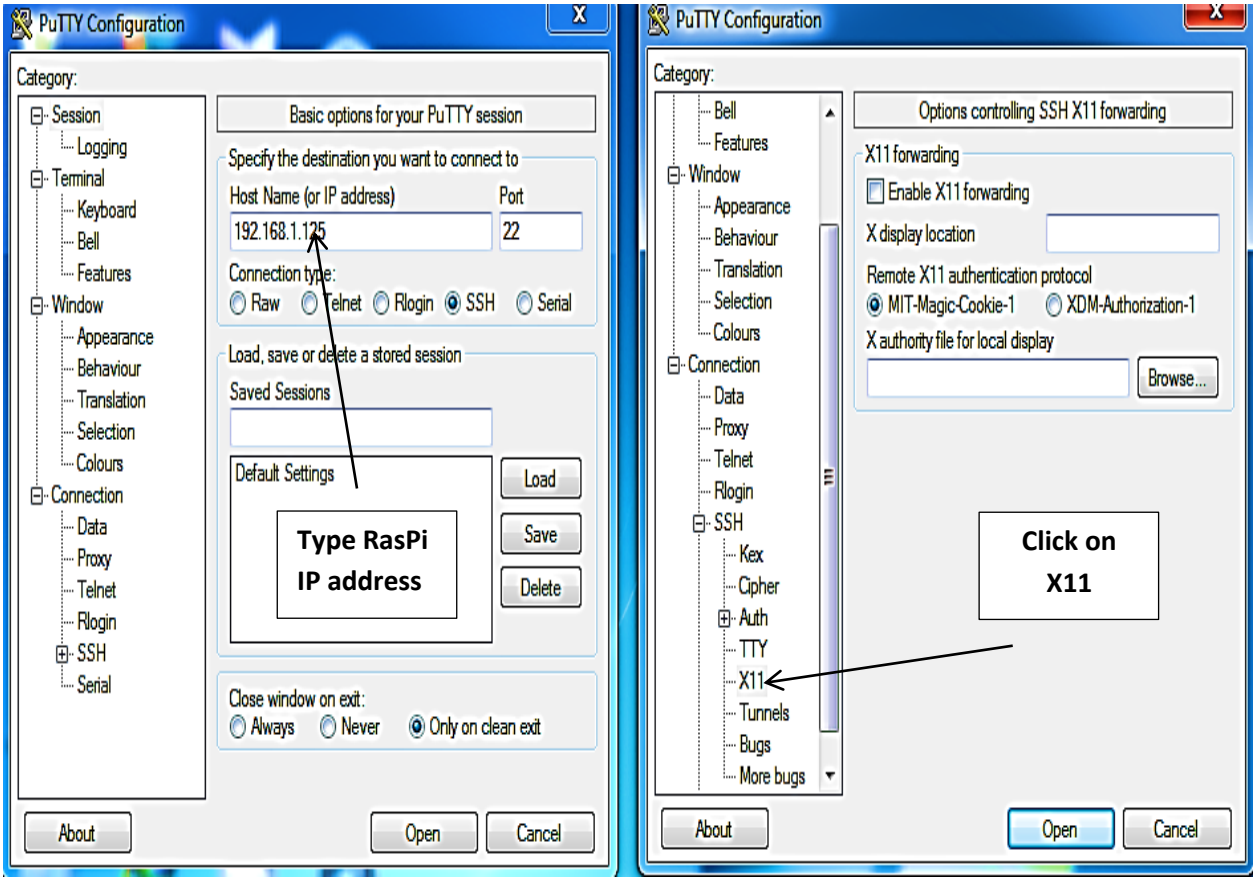
http://www.chiark.greenend.org.uk/~sgtatham/putty/

Step2:

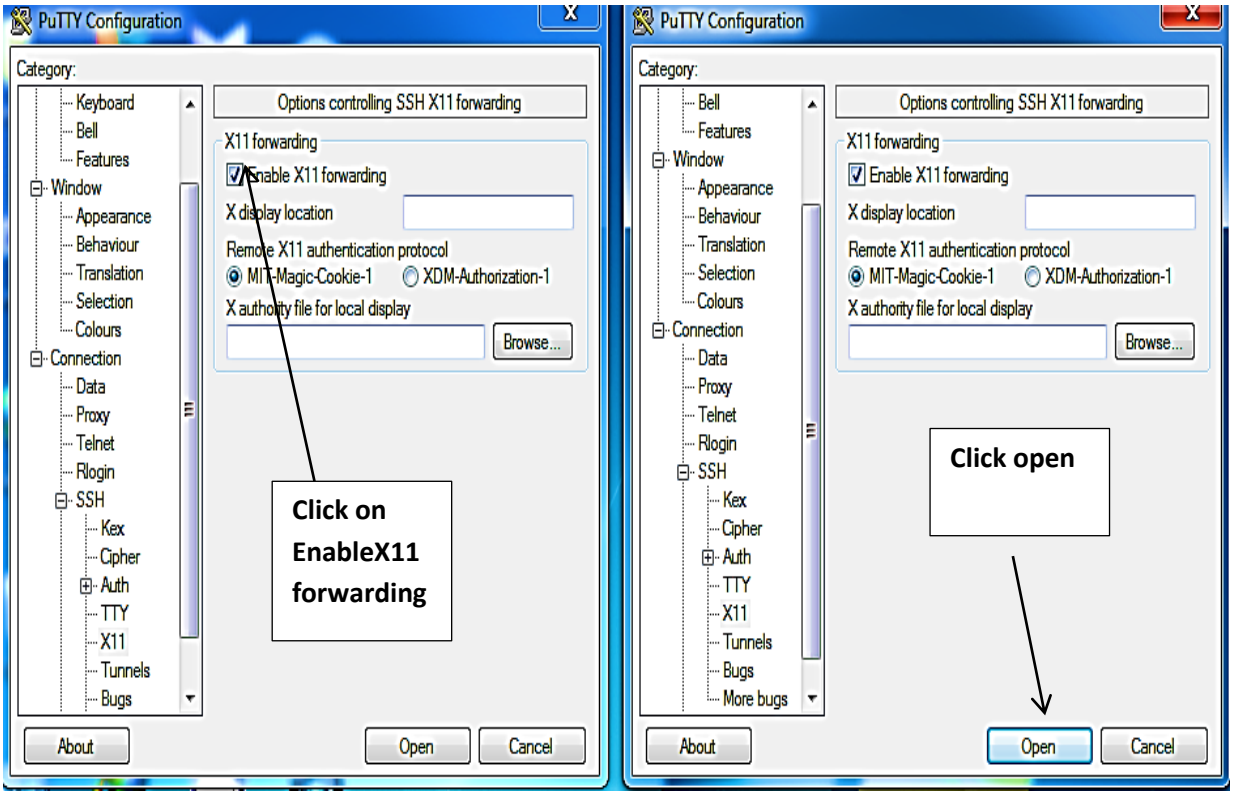
Download and install Xming using below site.

http://www.straightrunning.com/xmingnotes/

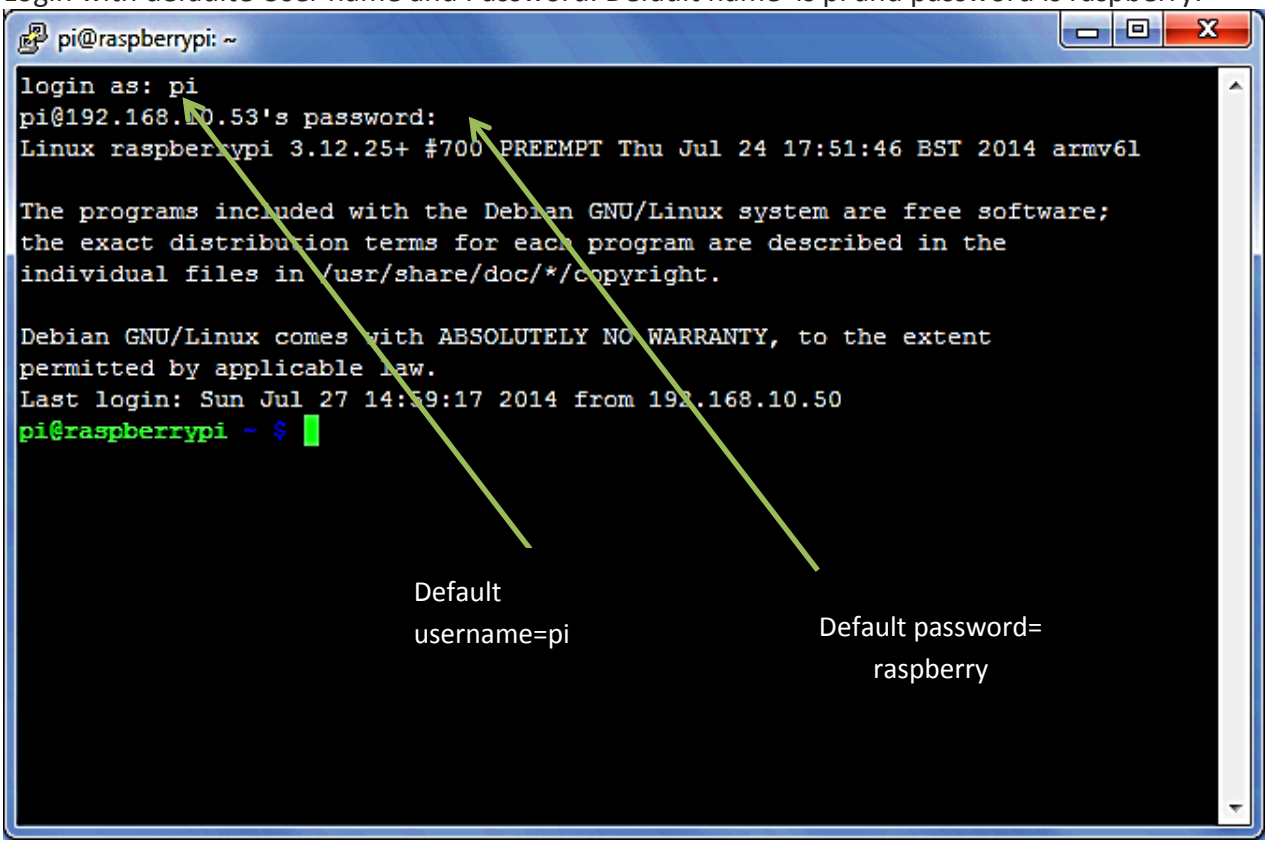
Step3:
Configure PuTTY as shown below figures


PuTTY Configuration
Category:
Session
Logging
Terminal
Keyboard
Bell
Features
Window
Appearance
Behaviour
Translation
Selection
Colours
Connection
Data
Proxy
Telnet
Rlogin
SSH
Serial
Basic options for your PuTTY session
Specify the destination you want to connect to
Host Name (or IP address)
Port
192.168.1.125
22
Connection type:
Raw
Telnet
Rlogin
SSH
Serial
Load, save or delete a stored session
Saved Sessions
Default Settings
Load
Save
Delete
Type RasPi IP address
Close window on exit:
Always
Never
Only on clean exit
About
Open
Cancel
PuTTY Configuration
Category:
Bell
Features
Window
Appearance
Behaviour
Translation
Selection
Colours
Connection
Data
Proxy
Telnet
Rlogin
SSH
Kex
Cipher
Auth
TTY
X11
Tunnels
Bugs
More bugs
Options controlling SSH X11 forwarding
X11 forwarding
Enable X11 forwarding
X display location
Remote X11 authentication protocol
MIT-Magic-Cookie-1
XDM-Authorization-1
X authority file for local display
Browse...
Click on X11
About
Open
Cancel

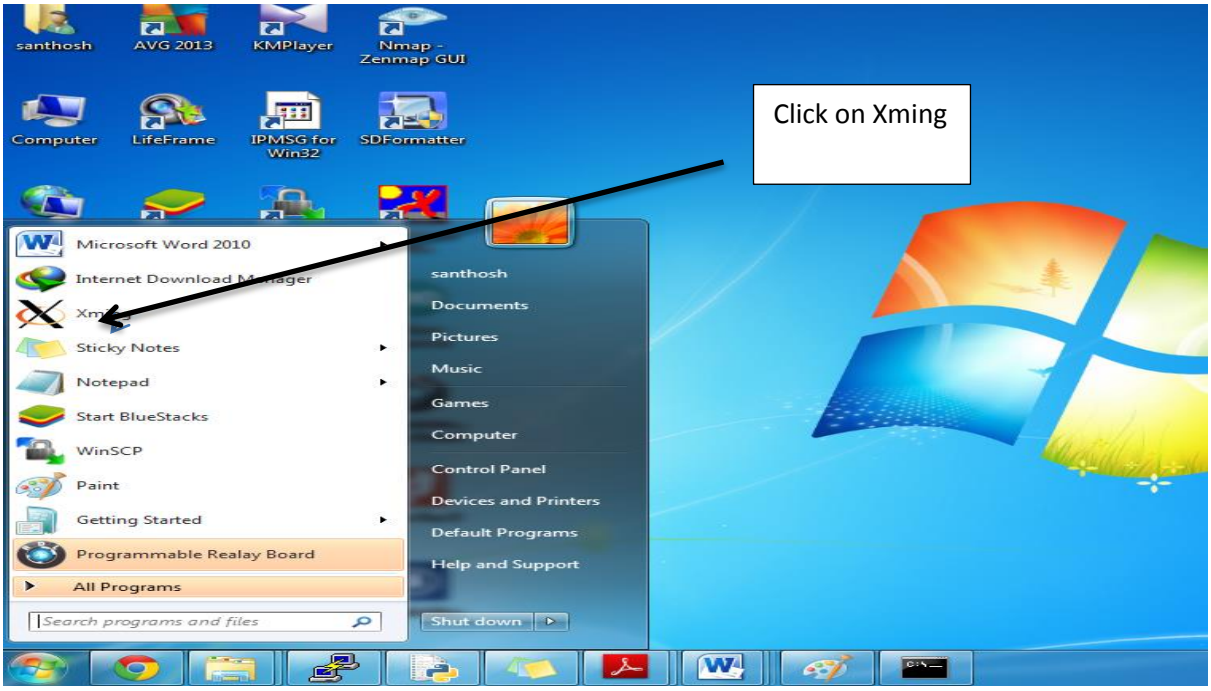
as


PuTTY Configuration
Category:
Keyboard
Bell
Features
Window
Appearance
Behaviour
Translation
Selection
Colours
Connection
Data
Proxy
Telnet
Rlogin
SSH
Kex
Cipher
Auth
TTY
X11
Tunnels
Bugs
Options controlling SSH X11 forwarding
X11 forwarding
Enable X11 forwarding
X display location
Remote X11 authentication protocol
MIT-Magic-Cookie-1
XDM-Authorization-1
X authority file for local display
Browse...
Click on EnableX11 forwarding
About
Open
Cancel
PuTTY Configuration
Category:
Bell
Features
Window
Appearance
Behaviour
Translation
Selection
Colours
Connection
Data
Proxy
Telnet
Rlogin
SSH
Kex
Cipher
Auth
TTY
X11
Tunnels
Bugs
More bugs
Options controlling SSH X11 forwarding
X11 forwarding
Enable X11 forwarding
X display location
Remote X11 authentication protocol
MIT-Magic-Cookie-1
XDM-Authorization-1
X authority file for local display
Browse...
Click open
About
Open
Cancel

Step 4:
Login with defaulte User name and Password. Default name is pi and password is raspberry.

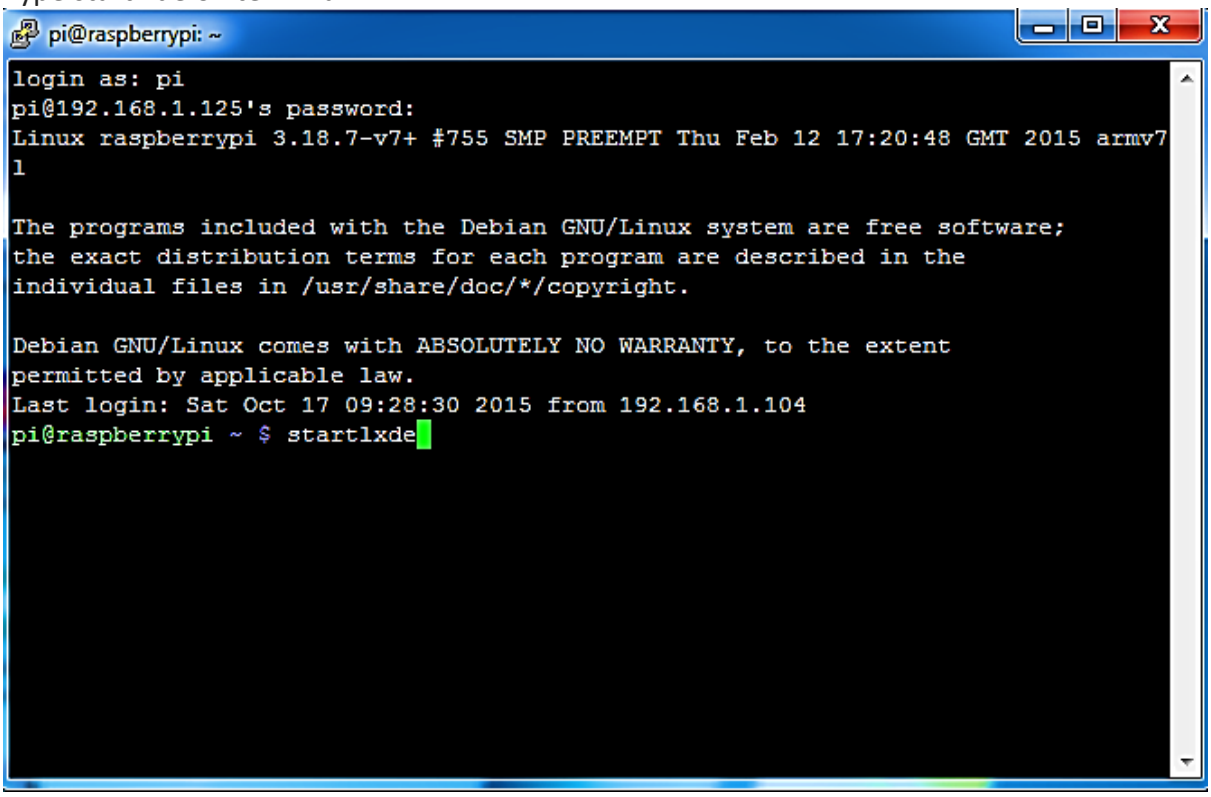


Step5:
Start xming

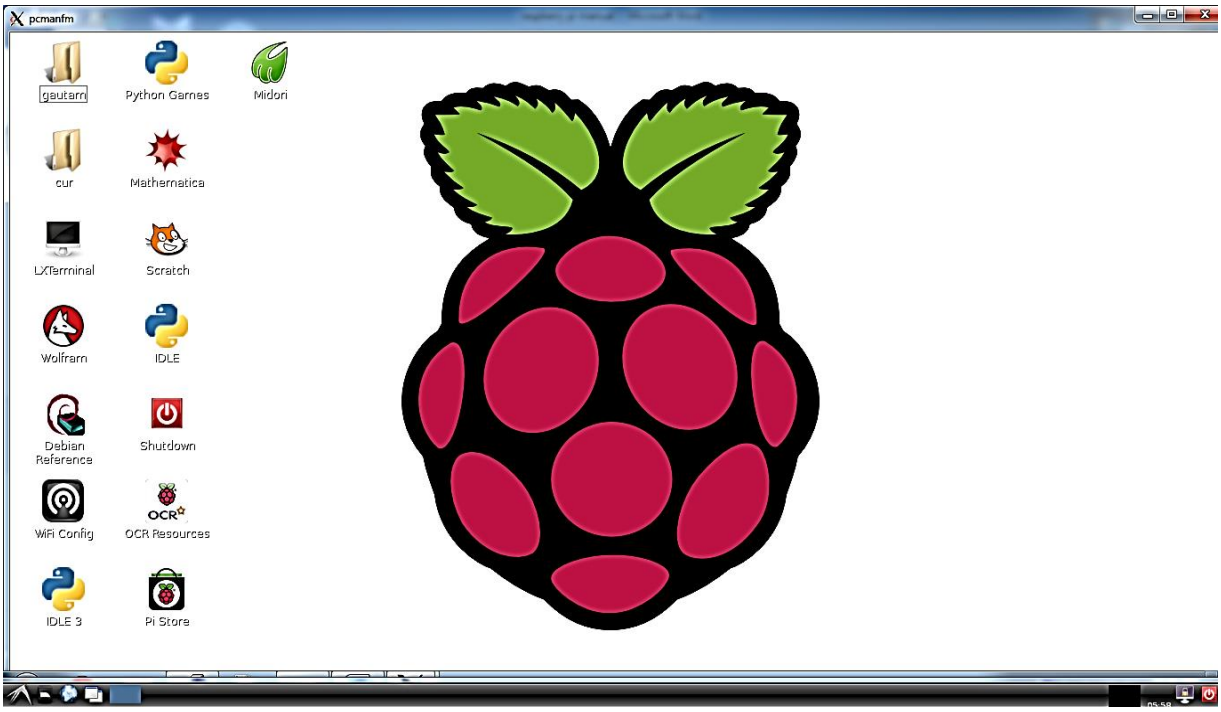


Step6:

Type startlxde on terminal



After pressing enter we can see Raspberry Pi window like this


pcmanfm
gautam
Python Games
Midori
cur
Mathematica
LXTerminal
Scratch
Wolfram
IDLE
Debian Reference
Shutdown
Wifi Config
OCR Resources
IDLE 3
Pi Store

ACCESSING RASPBERRY FILES IN WINDOWS MACHINE:

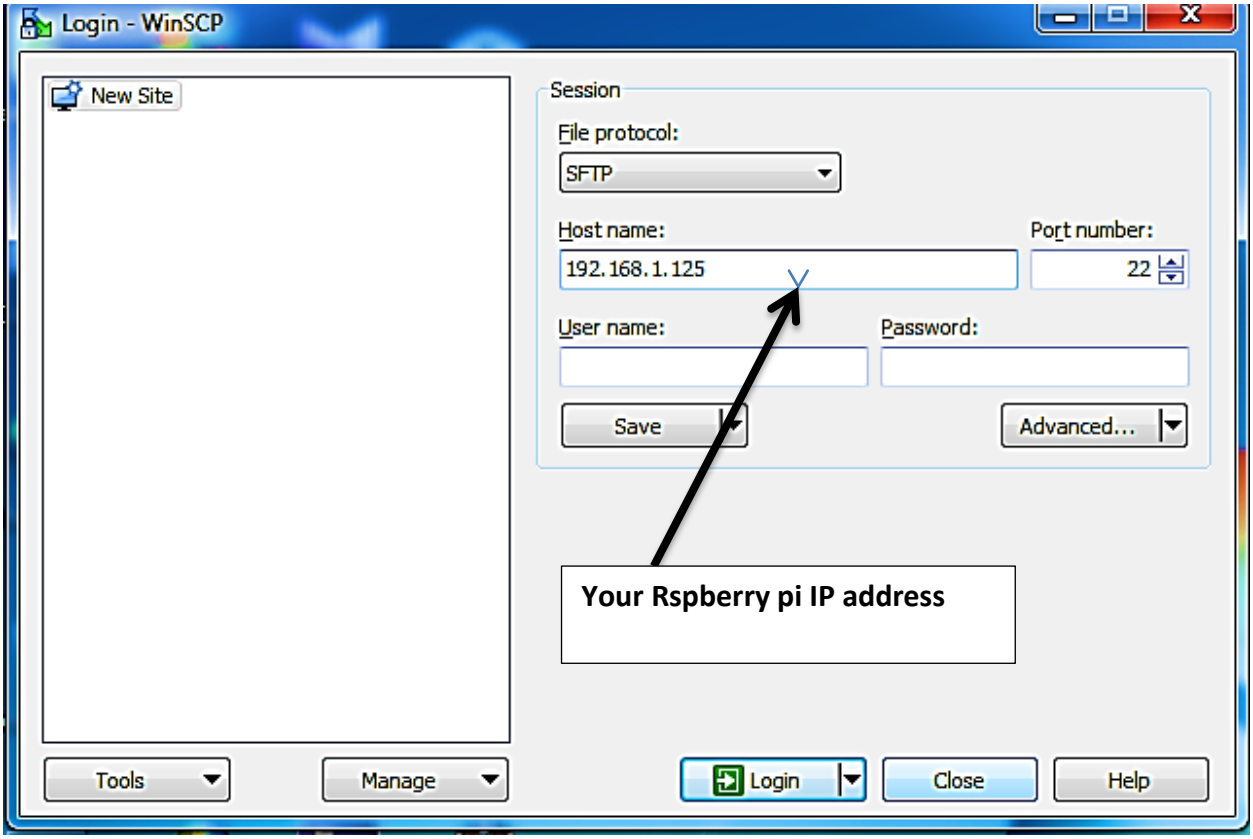
If we want to copy ,delet files from RasPi or from windows then its possible by using spftware called winSCP. Below illustrates the procedure.

Step1:
Download and install winSCP software here

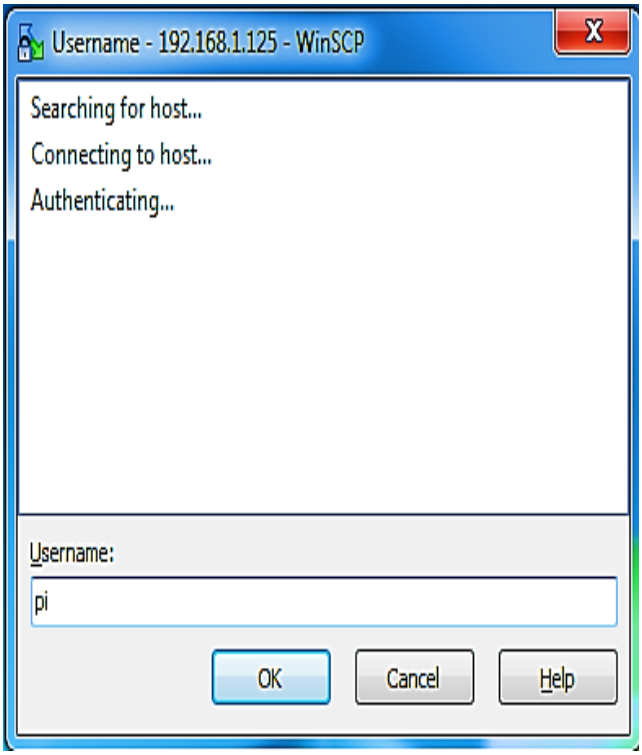
https://winscp.net/eng/download.php#download2

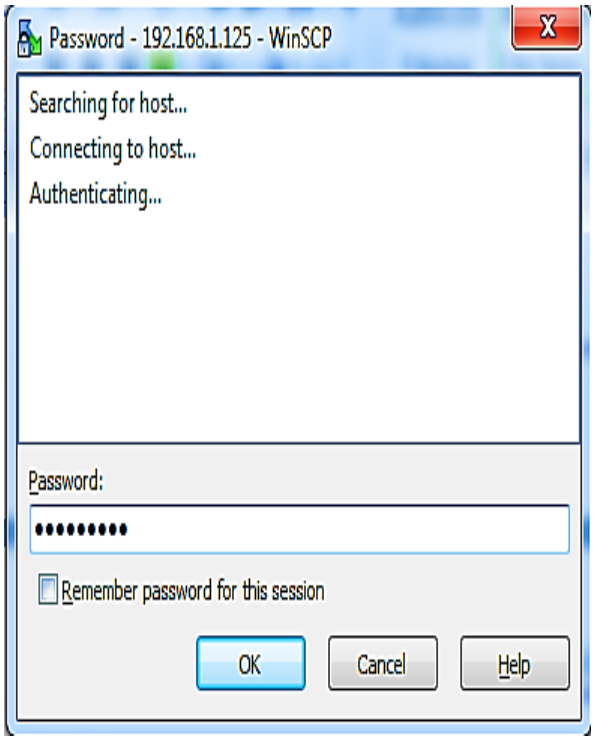
Step2:

Open winSCP and type RasPi IP address.



Step2:
Next it asks for user name and password. Type Raspberry pi default username and password i.e pi and raspberry

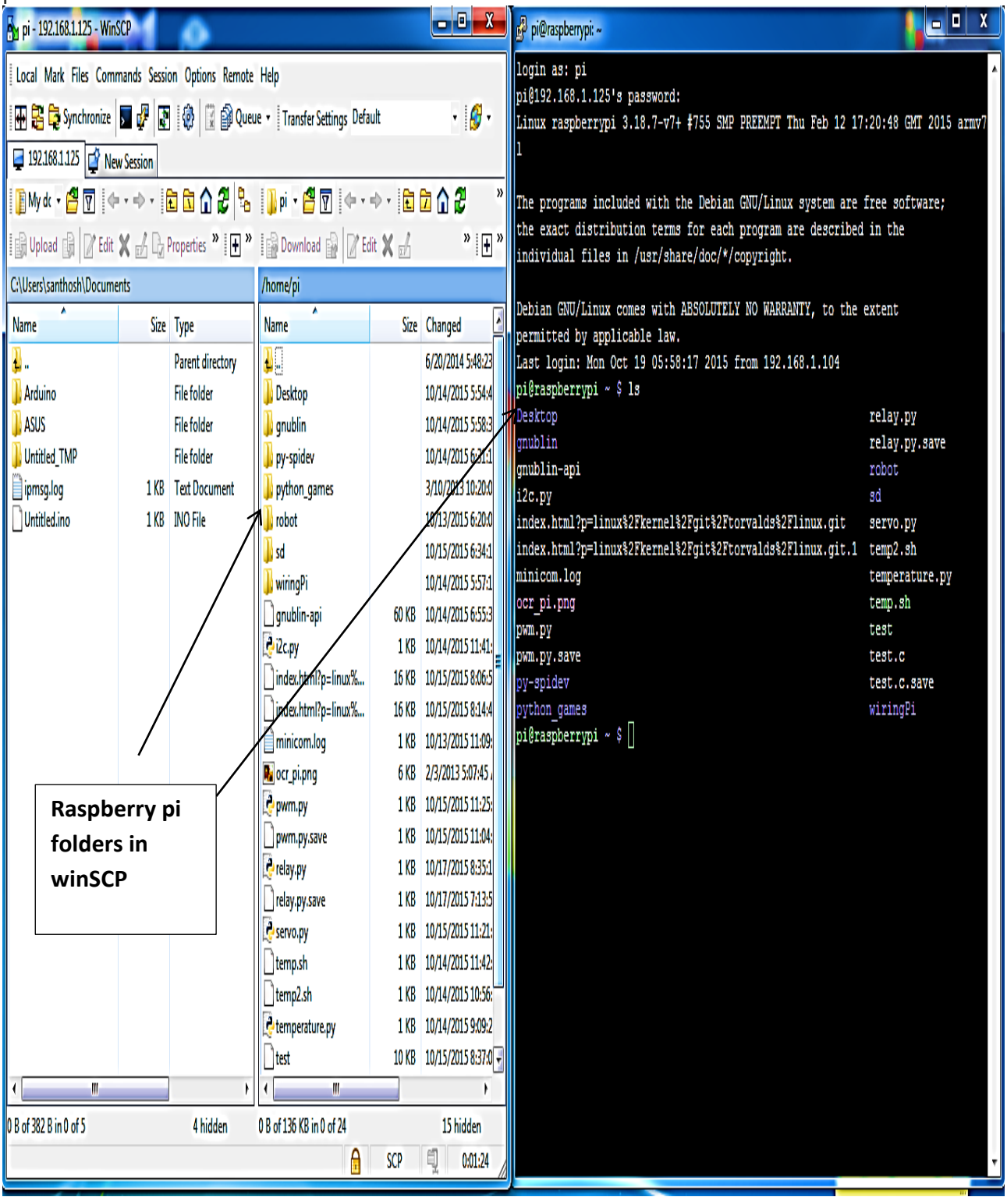




Step3:

We can see the Raspberry pi folders in winSCP .

pi


pi - 192.168.1.125 - WinSCP
Local Mark Files Commands Session Options Remote Help
Synchronize
Queue
Transfer Settings Default
192.168.1.125
New Session
Upload
Edit
Properties
Download
Edit
C:\Users\santhosh\Documents
/home/pi
Name
Size
Type
Changed
Parent directory
Arduino
File folder
ASUS
File folder
Untitled_TMP
File folder
ipmsg.log
1 KB
Text Document
Untitled.ino
1 KB
INO File
Desktop
gnublin
py-spidev
python_games
robot
sd
wiringPi
gnublin-api
60 KB
i2c.py
1 KB
index.html?p=linux%...
16 KB
index.html?p=linux%...
16 KB
minicom.log
1 KB
ocr_pi.png
6 KB
pwm.py
1 KB
pwm.py.save
1 KB
relay.py
1 KB
relay.py.save
1 KB
servo.py
1 KB
temp.sh
1 KB
temp2.sh
1 KB
temperature.py
1 KB
test
10 KB
0 B of 382 B in 0 of 5
4 hidden
0 B of 136 KB in 0 of 24
15 hidden
SCP
0:01:24
Raspberry pi folders in winSCP
pi@raspberrypi: ~
login as: pi
pi@192.168.1.125's password:
Linux raspberrypi 3.18.7-v7+ #755 SMP PREEMPT Thu Feb 12 17:20:48 GMT 2015 armv7l
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: Mon Oct 19 05:58:17 2015 from 192.168.1.104
pi@raspberrypi ~ $ ls
Desktop
gnublin
gnublin-api
i2c.py
index.html?p=linux%2Fkernel%2Fgit%2Ftorvalds%2Flinux.git
index.html?p=linux%2Fkernel%2Fgit%2Ftorvalds%2Flinux.git.1
minicom.log
ocr_pi.png
pwm.py
pwm.py.save
py-spidev
python_games
relay.py
relay.py.save
robot
sd
servo.py
temp2.sh
temperature.py
temp.sh
test
test.c
test.c.save
wiringPi
pi@raspberrypi ~ $

BASIC LINUX COMMANDS:

Raspberry Pi uses Linux as its standard operating system, which is an operating system loosely based on the Unix operating system. It has always been a free and open source, written in C programming language and was originally designed to run on Intel's x86-based computers. To interact with RasPi we have to use command-Line Interface, where we are going to be doing a lot of work.

To get around in the Linux CLI, we have to use the file system commands such as cd and ls. Commands to run the programs are run from the terminal as well. Some of the common Linux commands are listed in table below.

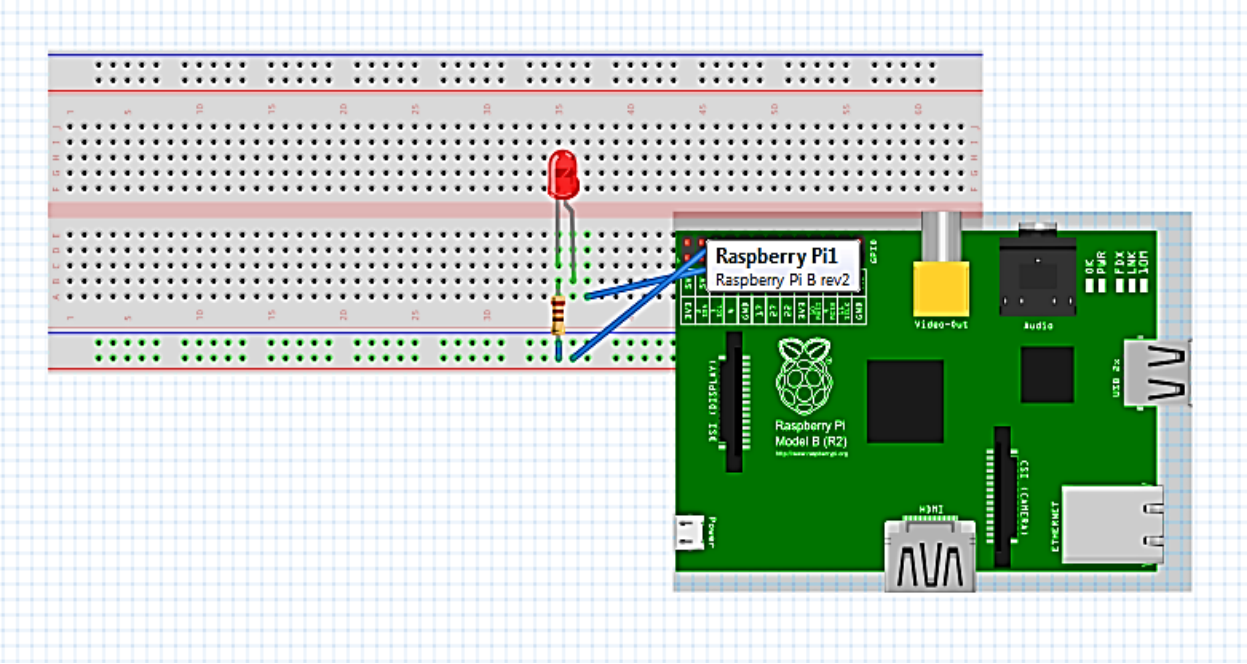
Command	Meaning
ls	List files in current directory
cd	Change directory
Pwd	Print working directory
rm *filename*	Remove *filename*
mkdir *directory name*	Make a directory with *directory name*
rmdir *directory name*	Remove empty directory
cat *textfile*	Display contents of *textfile* in the terminal
mv *oldfile newfile*	Move *oldfile* to *newfile*
cp *oldfile newfile*	Copy *oldfile* to *newfile*
Man *command*	Display manual of *command*
Date	Reads systems date/time
Echo	Echo types back in the terminal
Grep	Search program that uses regular expressions
Sudo	Perform as root user
./program	Run *program*
Exit	Quit terminal session

BLINKING LED:

The first experiment usually in hardware is blinking of an led. Below procedure shows how to blink an led using Raspberry pi.

Component required:
Raspberry pi
LED
Resistor-220 ohm
Breadboard
Wires

Circuit :



Program:

```
#santhosh SJEC
gpio mode  0 out    (1)
while (true);    (2)
do
gpio write 0 0 (3)
sleep 0.2  (4)
gpio write 0 1  (5)
sleep 0.2  (6)
```

done **(7)**

(1) Here mode 0 of raspberry gpio that is pin no 17 where the LED is connected is selected as an output

(2) Next is a while loop which is an infinite loop

(3) gpio write 0 0 this command writes 0 to mode 0 pin that is writes 0 to pin no 17 which turns LED off

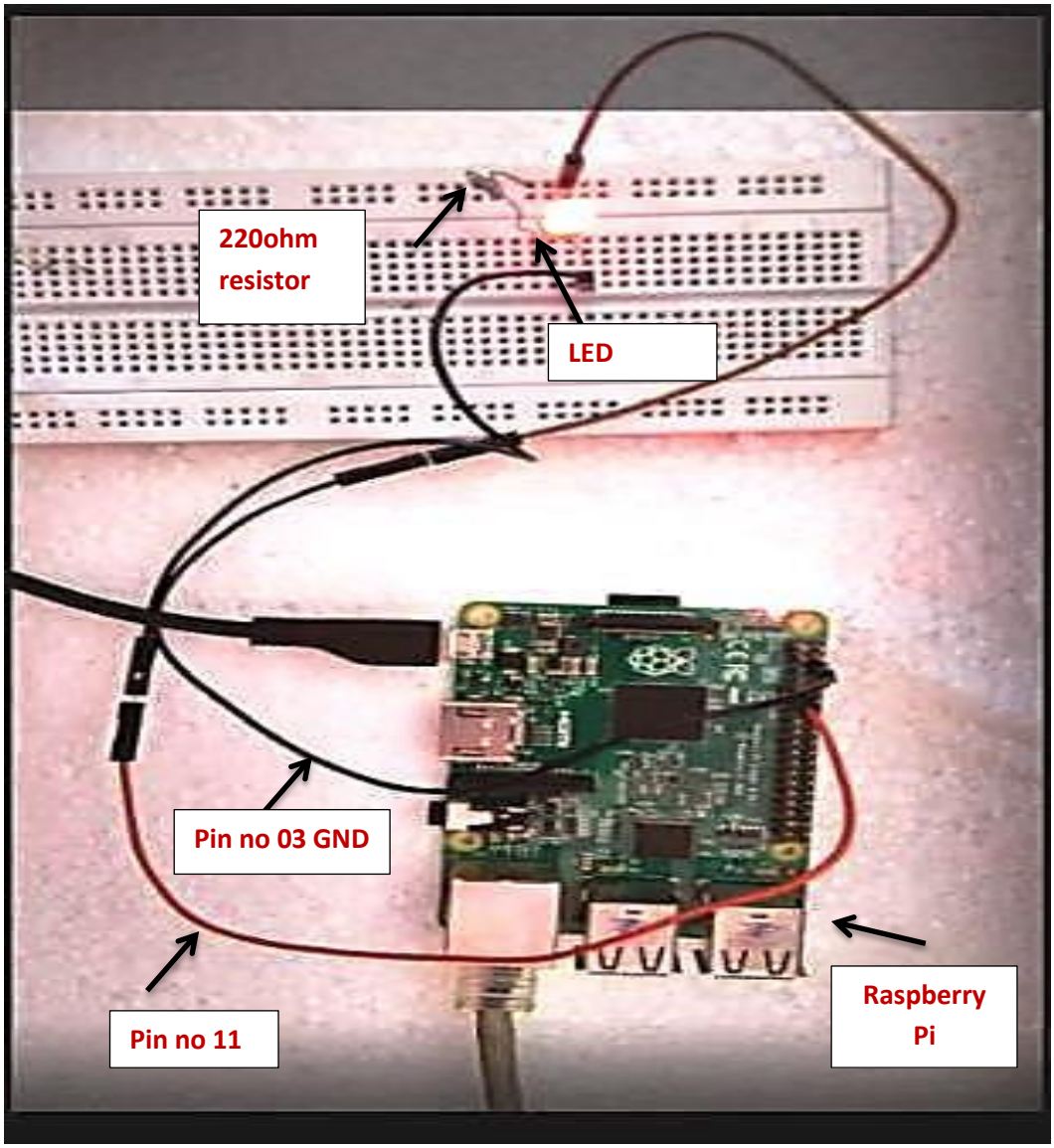
(4) sleep 0.2 is a delay function that is LED is off for the period of 2ms

(5) gpio write 0 1 this command writes 1 to mode 0 pin that is writes 1 to pin no 17 which turns LED on

(6) sleep 0.2 sleep 0.2 is a delay function that is LED is off for the period of 2ms

Since this is inside while loop process is repeated forever

Output:

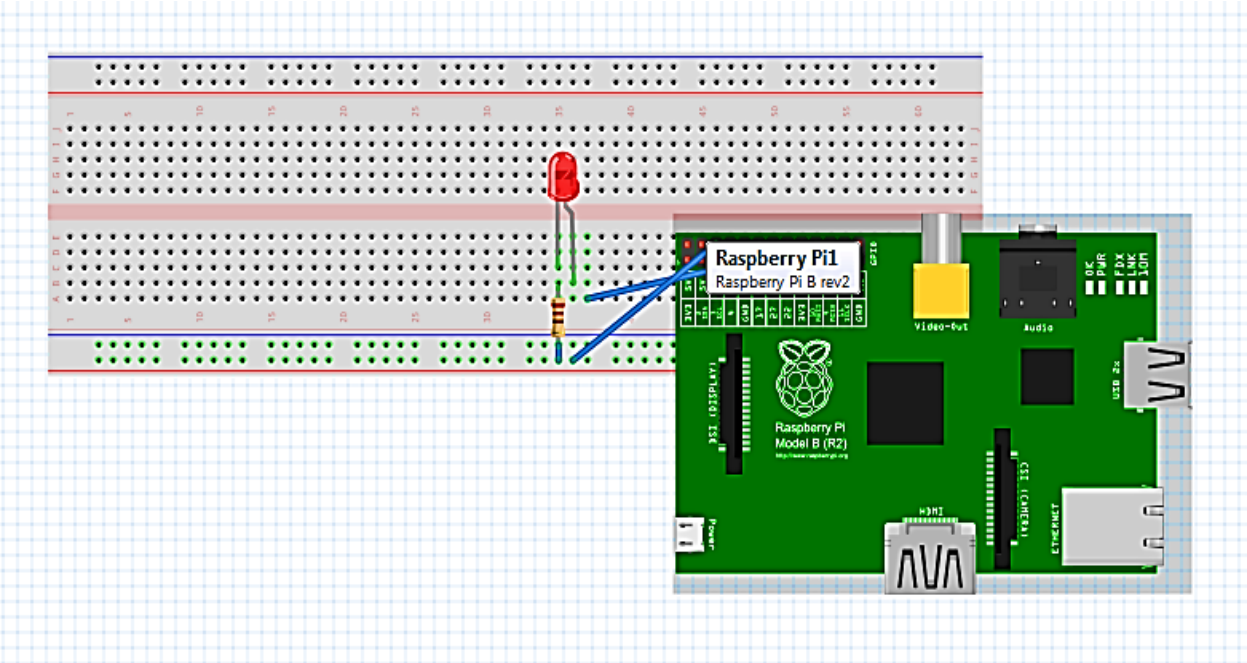


DIMMING LIGHT USING PWM:

PWM is a technique for controlling power. Below experiment shows how this technique is used to control the brightness of LED using python code.

Component requires:
Raspberry pi
LED
Resistor-220 ohm
Breadboard
Wires

Circuit:



Program :
PWM program for controlling led

```
import RPi.GPIO as GPIO          #importing GPIO librarie
import time                              #importing time
GPIO.setmode(GPIO.BCM)
GPIO.setup(18,GPIO.OUT)                 #LED is connected to pin no 11
p=GPIO.PWM(18,50)                #this initialises frequency of 50Hz
p.start(10)                       # starting PWM with duty cycle of 10


while True:
      for i in range(100):
            p.ChangeDutyCycle(i)
```

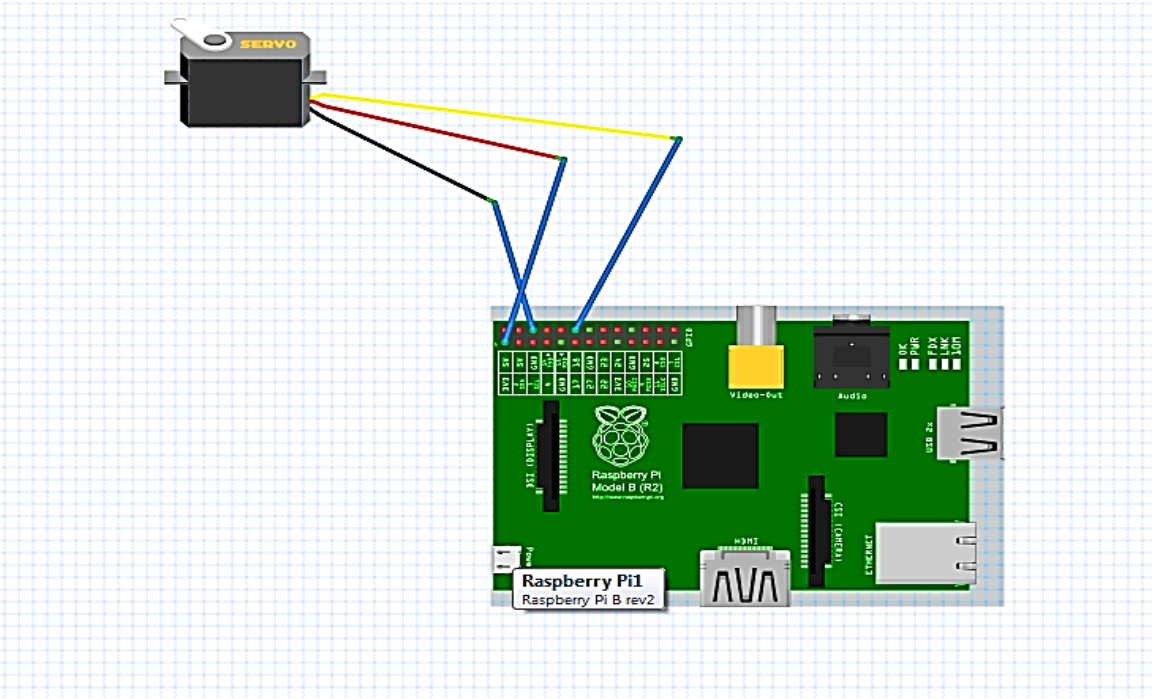
```
            print i
            time.sleep(0.02)
            print"**************"
        for j in range(100):
            p.ChangeDutyCycle(100-j)
            time.sleep(0.02)

            print j
            print "===================="

p.stop()

GPIO.cleanup()
```

SERVO MOTOR DIRECTION CONTROL USING PWM:

Components required:
Raspberry pi
Servo motor
Can buy from: http://researchdesignlab.com/index.php/robotics-kits/servo-motor.html

Wires

Circuit :



Program :

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

GPIO.setup(18,GPIO.OUT)

freq = 50
pwm = GPIO.PWM(18,freq)

LP = 0.75
RP = 2.5
MP = (RP-LP)/2+LP

PosList = [LP,RP,MP]
```

Importing GPIO libraries

Accessing GPIO pins as BCM mode

Setting it as OUTPUT

Initializing PWM with frequency

Setting position

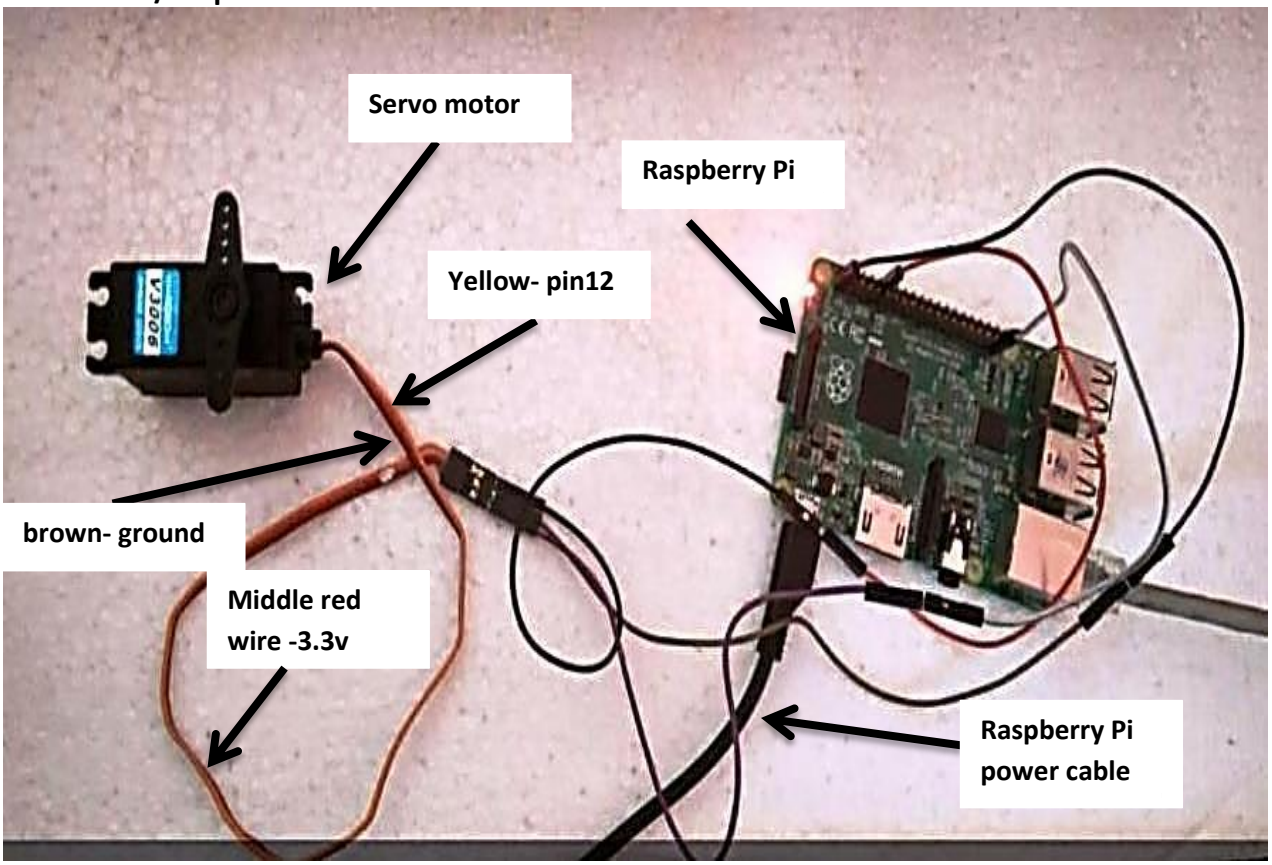
```
percycle= 1000/freq
while True:
      for i in range(3):
            for pos in PosList:
                  DC= pos*100/percycle
                  print "Position" + str(pos)
                  print "DutyCycle" + str(DC) +"%"
                  print ""
                  pwm.start(DC)
                  time.sleep(0.5)

pwm.stop()

GPIO.cleanup
```

Setting duty cycle

Starting PWM motor will rotate now with delay of 0.5 sec

Stopping pwm motor will stop now

Connection/output:

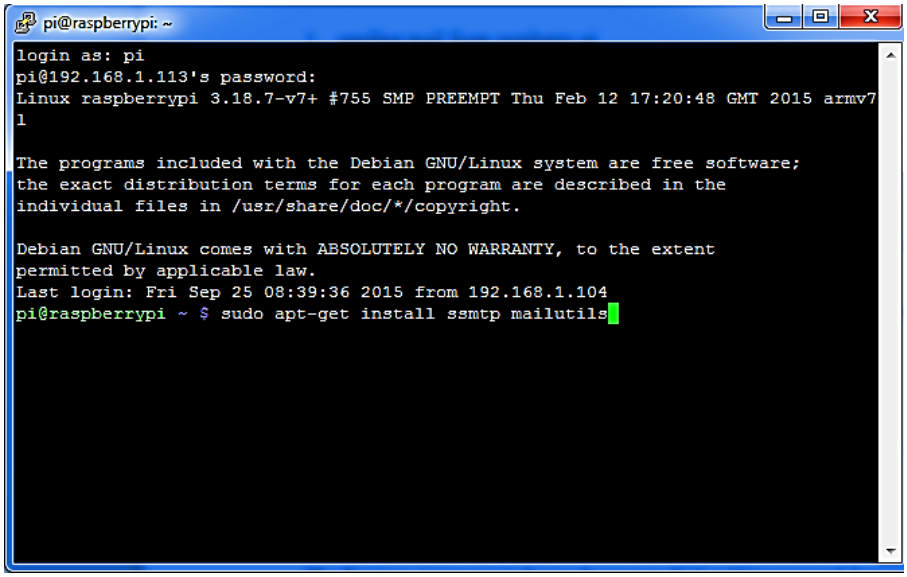


SENDING E-MAIL FROM RASPBERRY PI:

Step1: Sending mail from Raspberry Pi

To send mail from RasPi we have to install some important packages as follows. First, we have to download and install ssmtp package which is used to send e-mails, stands for simple mail transport protocol. We also have to install mailutils which is setup libraries for handling e-mails, which can be done by using the command, shown in fig 6 (a)

"sudo apt-get install ssmtp mailutils"



Once the installation is completed, we need to edit the ssmtp configuration files by using command

"sudo nano etc/ssmtp/ssmtp.conf".

In configuration file first we have to set mail hub, here I have set mail hub to gmail.com with port no 465, that is by writing

"**mailhub=smtp.gmail.com:465".**

Next we have to add following three lines

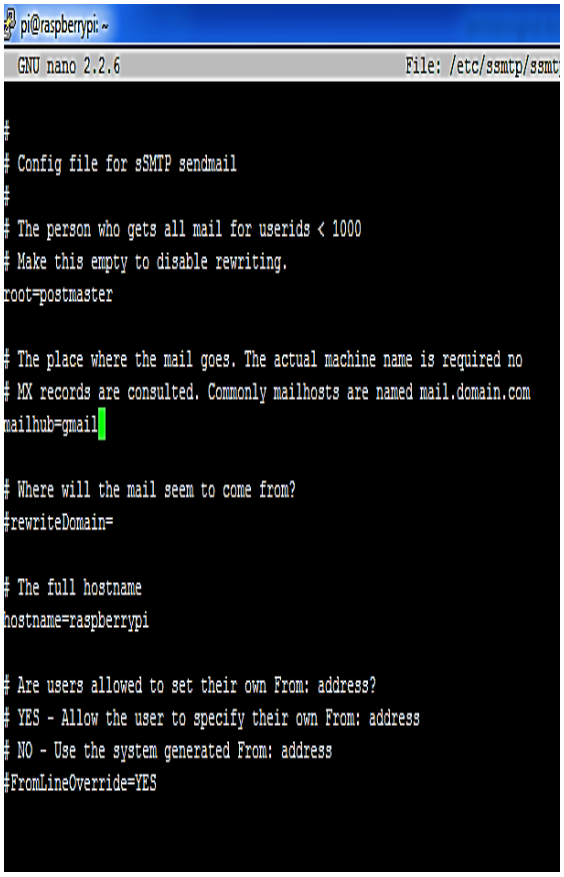
AuthUser = santhuraspberrygmail.com

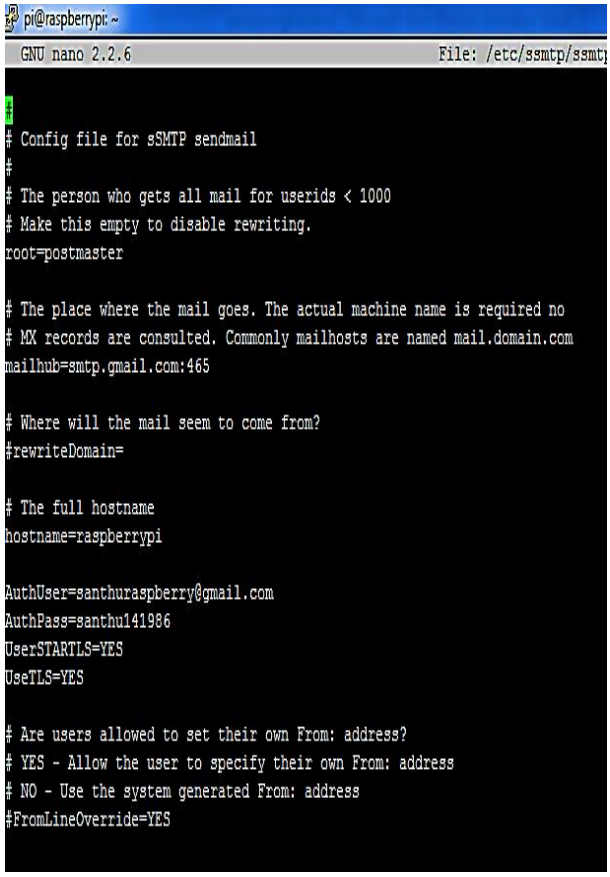
AuthPass= santhu141986

UserSTARTTLS=YES

UseTLS=YES

The first two statements indicate the user mail id and his password, here mail id is santhuraspberry@gmail.com and password is santh141986. The next two statements are for encryption. Figure below shows after configuration



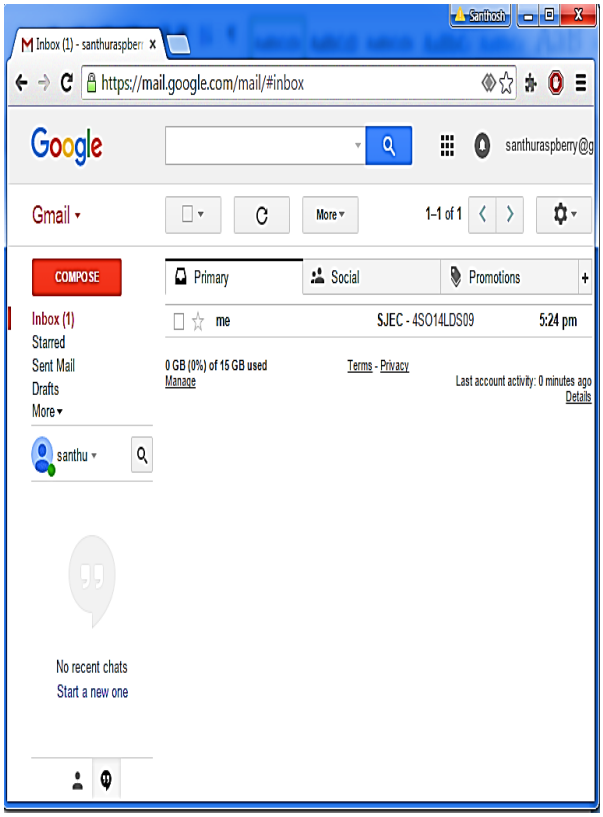


After all these settings we are now ready to send a mail. The mail can be sent by using following Linux script

“echo "4SO14LDS09" | mail -s "SJEC" santhuraspberry@gmail.com ”

Here first part contains the body of the mail i.e. “4SO14LDS09”, second part is the subject, i.e. “SJEC” and the third part is the address of the e-mail will be sent to.


Inbox - santhuraspberry@
https://mail.google.com/mail/#inbox
Google
Gmail
COMPOSE
Inbox
Starred
Sent Mail
Drafts
More
Primary
Social
Promotions
Your Primary tab is empty.
Personal messages and messages that don't appear in other tabs will be shown here.
To add or remove tabs click inbox settings.
santhu
No recent chats
Start a new one
Account Created


Inbox (1) - santhuraspber
https://mail.google.com/mail/#inbox
Google
Gmail
COMPOSE
Inbox (1)
Starred
Sent Mail
Drafts
More
Primary
Social
Promotions
1-1 of 1
me
SJEC - 4SO14LDS09
5:24 pm
0 GB (0%) of 15 GB used
Manage
Terms - Privacy
Last account activity: 0 minutes ago
Details
santhu
No recent chats
Start a new one

USING MJPEG-STREAMER TO STREAM VIDEO OVER HTTP:

This project illustrates how we can connect our webcam to the internet and perform live video streaming. To do this what we required is mjpeg_streamer program that gets the MJPG data from V4L2 and send it through an HTTP session. MJPG-streamer, is a command line application that copies JPG-frame from single input plugin to multiple output plugins. It can be used to stream a JPEG over an IP based network from the webcam to the viewer like a Firefox, etc. Mjpeg streamer automatically generates a set of html pages that illustrates different methods to stream the video over our browser.

Step 1: Installing packages needed for MJPG streamer

The following command installs three libraries that MJPG streamer uses

$ sudo apt-get install libjpeg8-dev imagemagick libv4l-dev

Here the first package being installed is **libjpeg8-dev**, which is used to handle JPEG files. Next package is the **imagemagick,** software to create, edit, compose or convert bitmap images. It is also used to resize,
flip, rotate, distort and transform images. Last packages is the **libv4l-dev** is a collection of libraries which adds a layer top of V4l2 devices. The purpose of this layer is it prevents writing separate code for different devices in the same class

Step 2: Creating symbolic link for videodev.h

The videodev.h is the header file that MJPG-streamer requires. The following command creates the symbolic link, which videodev.h with newer version videodev2.h.

ln -s /usr/include/linux/videodev2.h /usr/include/linux/videodev.h

Step 3: Downloading MJPG-Streamer

This step downloads source code for MJPG streamer, which is available at sourceforget.net.

$ wget <http://sourceforge.net/code-snapshots/svn/m/mj/mjpg-streamer/code/mjpg-streamer-code-182.zip>

Step 4: Unzip the MJPG-Streamer source code

The source code downloaded is a compressed zip file. It can be unzipped by the following command.

$ unzip mjpg-streamer-code-182.zip

Step 5: Build MJPG-Streamer

There are several plugins in MJPG - streamer, but only couple of them are needed to stream video. Them command below builds plugins what we needed.

$ cd mjpg-streamer-code-182/mjpg-streamer

$ make mjpg_streamer input_file.so output_http.so

Step 6: Copying needed files into system directories

The following command copies needed files into system directories. The first command copies mjpg-streamer to the local directories, second command copies libraries to the local directories and third command copies the files into a local directory.

$ sudo cp mjpg_streamer /usr/local/bin
$ sudo cp output_http.so input_file.so /usr/local/lib/
$ sudo cp -R www /usr/local/www

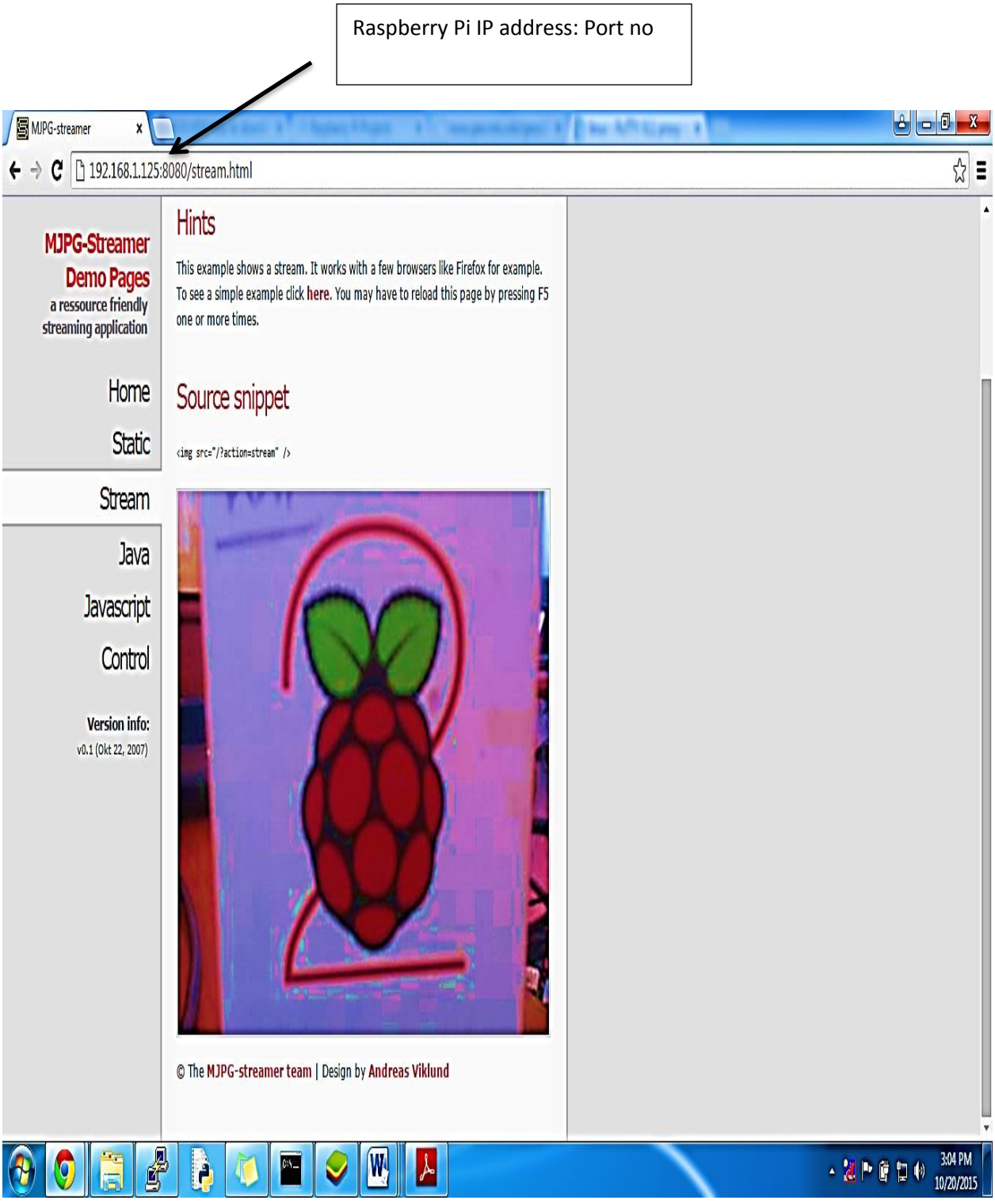
Step 7: Starting MJPG-streamer for the first time

The MJPG streamer can be started by using the following command

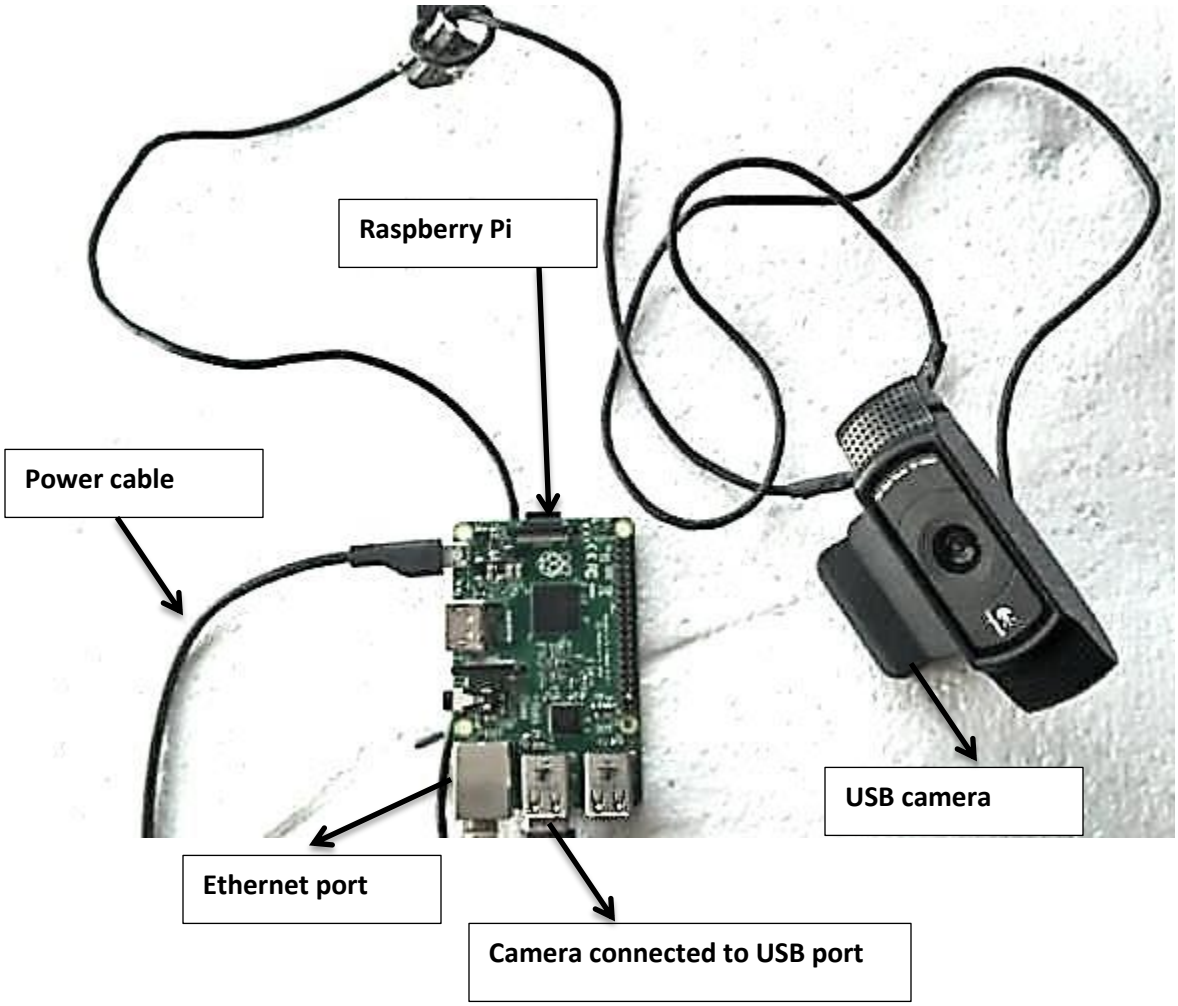
mjpg_streamer -i "/usr/local/lib/input_uvc.so -d /dev/video0 -r 640x480 -f 15" -o "/usr/local/lib/output_http.so -p 8080 -w /usr/local/www"

Here “input_uvc.so” captures the JPG frames from a connected webcam and output_http streams on the web with HTTP TCP port 8080. We can view our webcam streaming by opening browser with

http:// our IP address>:8080



Connection:


Raspberry Pi
Power cable
USB camera
Ethernet port
Camera connected to USB port

I2C :

It's a Inter-Integrated Circuit Bus. It's a simple bi-directional 2-wire bus for efficient inter-IC control. No specific wiring or connectors but just PCB tracks. (It's a two wire serial interface)Since a serial Interface it reduces cost of manufacturing of electronic product. „ Only two bus lines are required: a serial data line (SDA) and a serial clock line (SCL). Each device connected to the bus is software addressable by a unique address and simple master/slave relationships exist at all times; masters can operate as master-transmitters or as master-receivers. Below procedure shows how to configure Raspberry to use I2C.

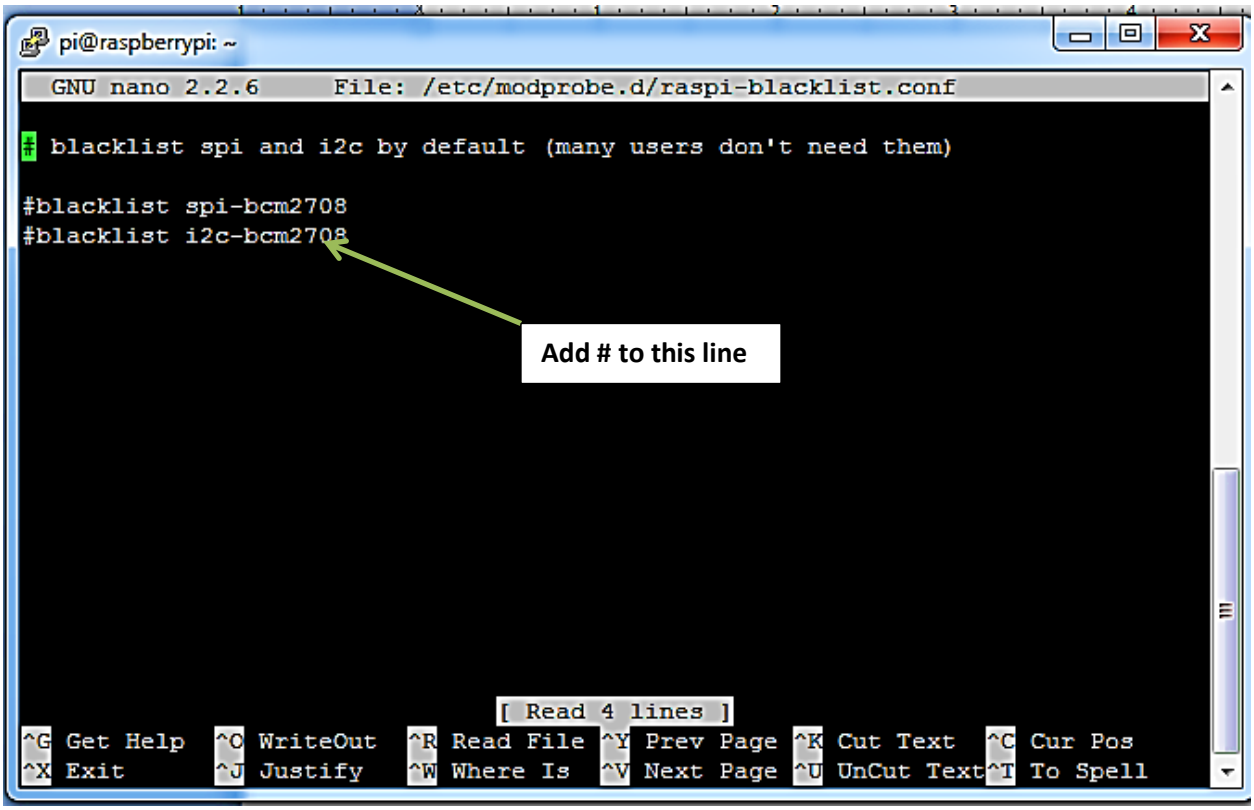
Step1:

Make sure your Raspberry Pi is connected to the internet when installing the drivers.

The new Raspbian distro already have the I2C driver installed but they are disabled by default. To enable it all you need to do is comment out a line by putting # in front. At the prompt type.

sudo nano /etc/modprobe.d/raspi-blacklist.conf

then add a # on the 3rd line.

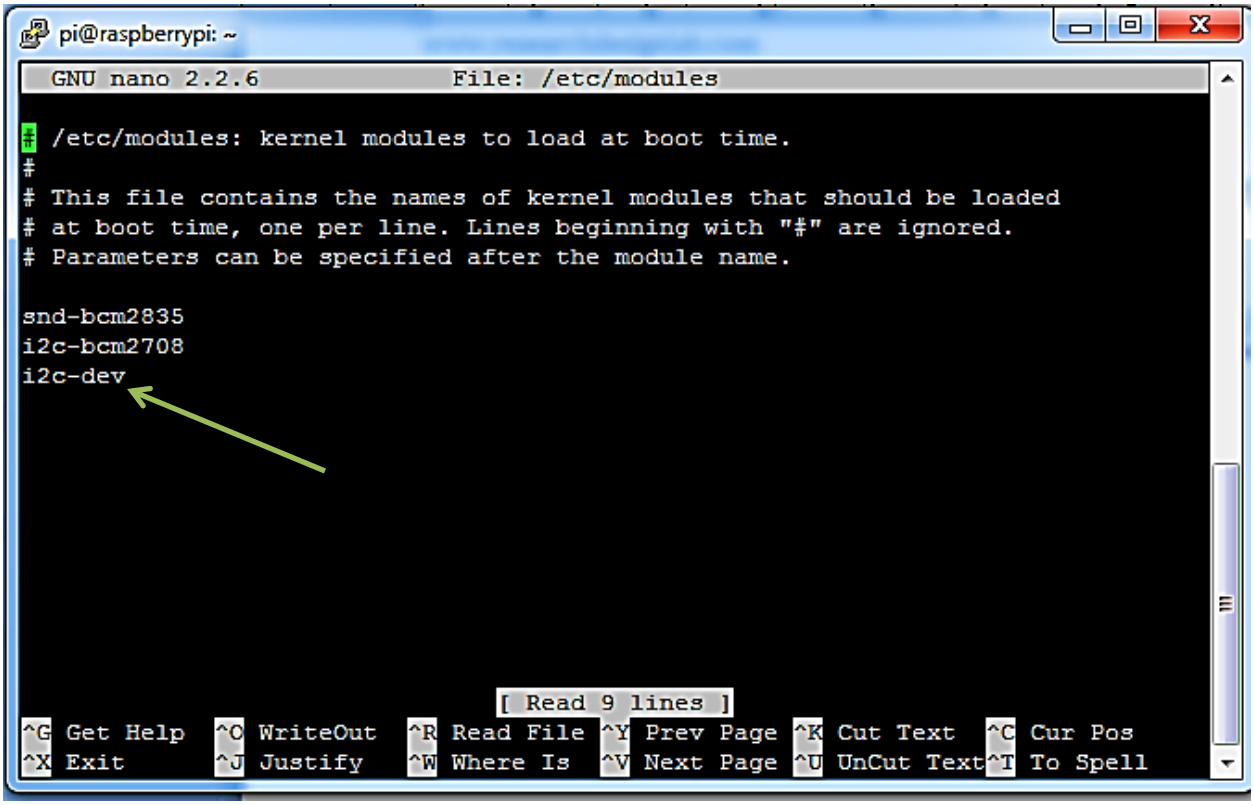


Step2:

Next edit the modules file by:

sudo nano /etc/modules

Add i2c-dev to a new line.



Step3:

Now install the i2c-tools package by:

sudo apt-get install i2c-tools

Step4:

Now add a new user to the i2c group:

sudo adduser pi i2c

Reboot the machine by:

sudo shutdown -r now

Step5:

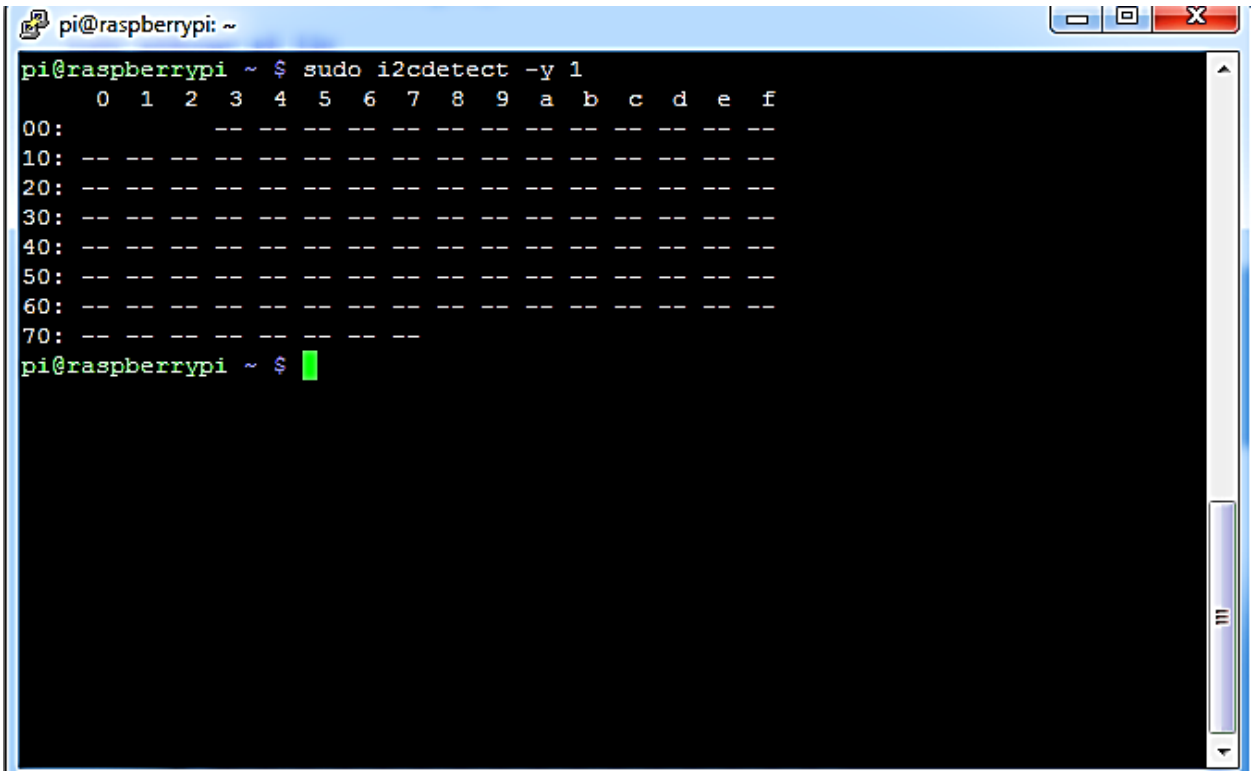
After the reboot test to see any device connected by:

sudo i2cdetect -y 0

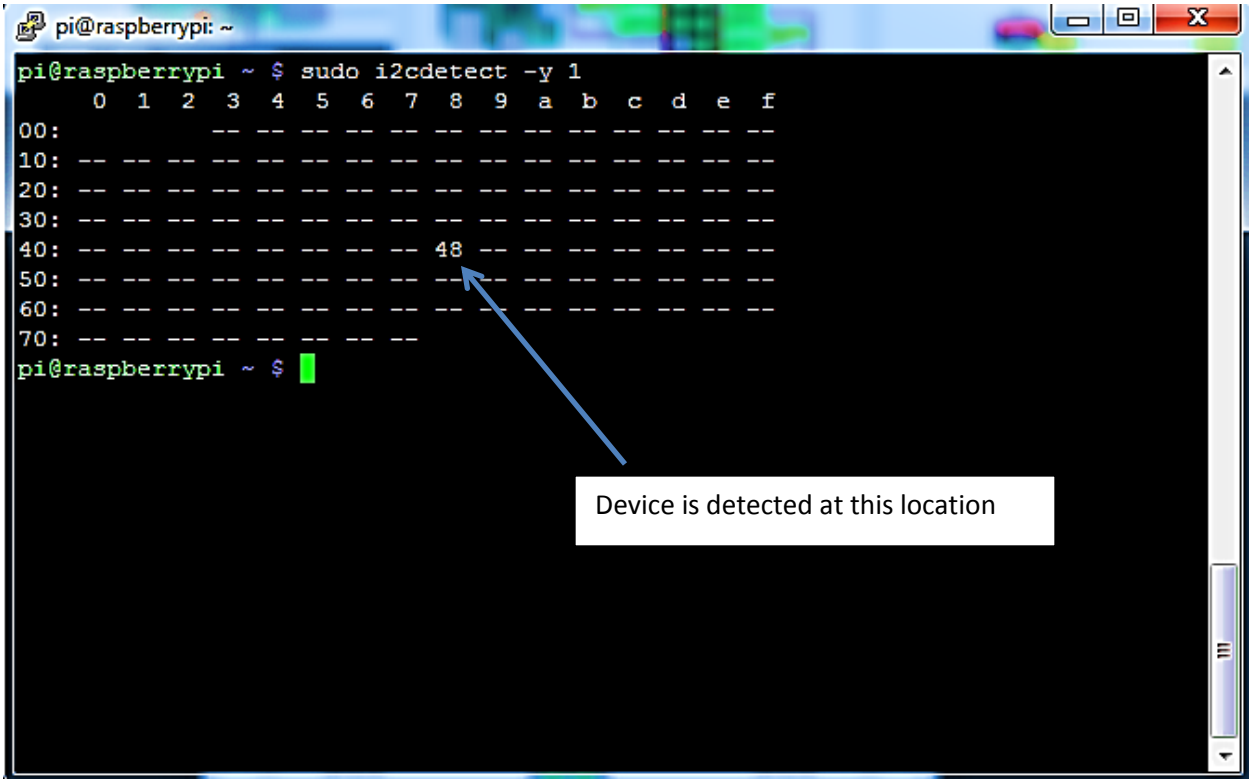
If your board is the Rev 2 type this:

sudo i2cdetect -y 1

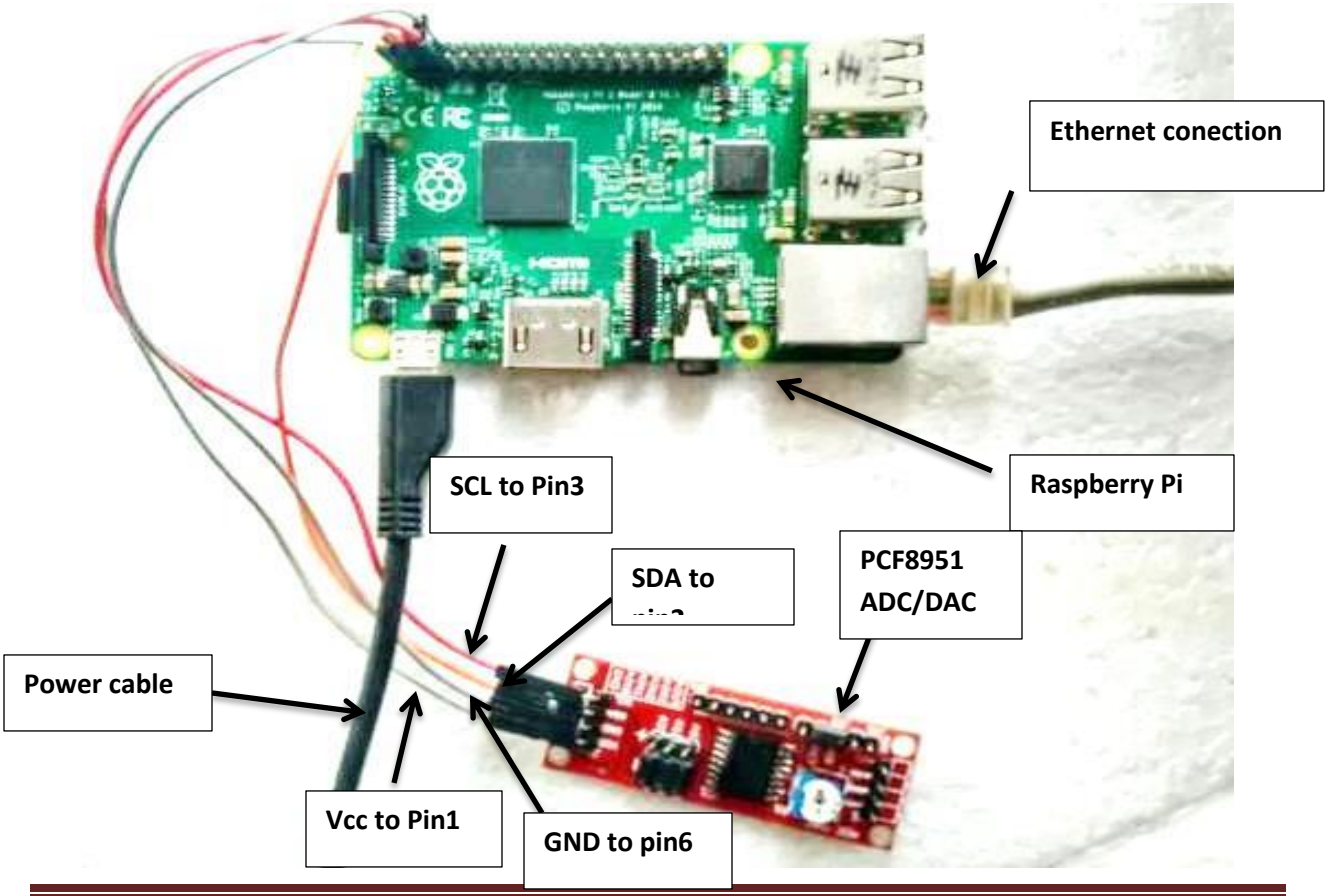
if no device is connected to I2C port then we will see something like this:



Now if any device is connected to I2C port then display will be something like this



Connections:

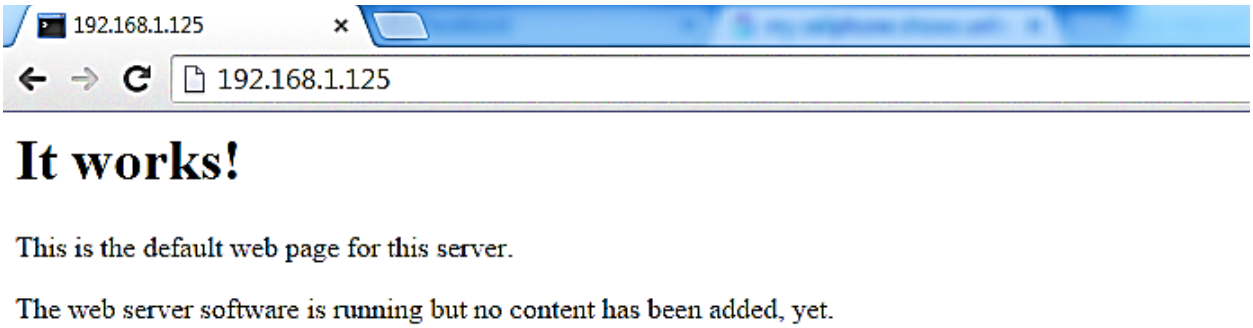


INSTALLING APACHE SERVER AND PHPADMIN:

Step1:
Type the following command on command line to install apache server on raspberry pi

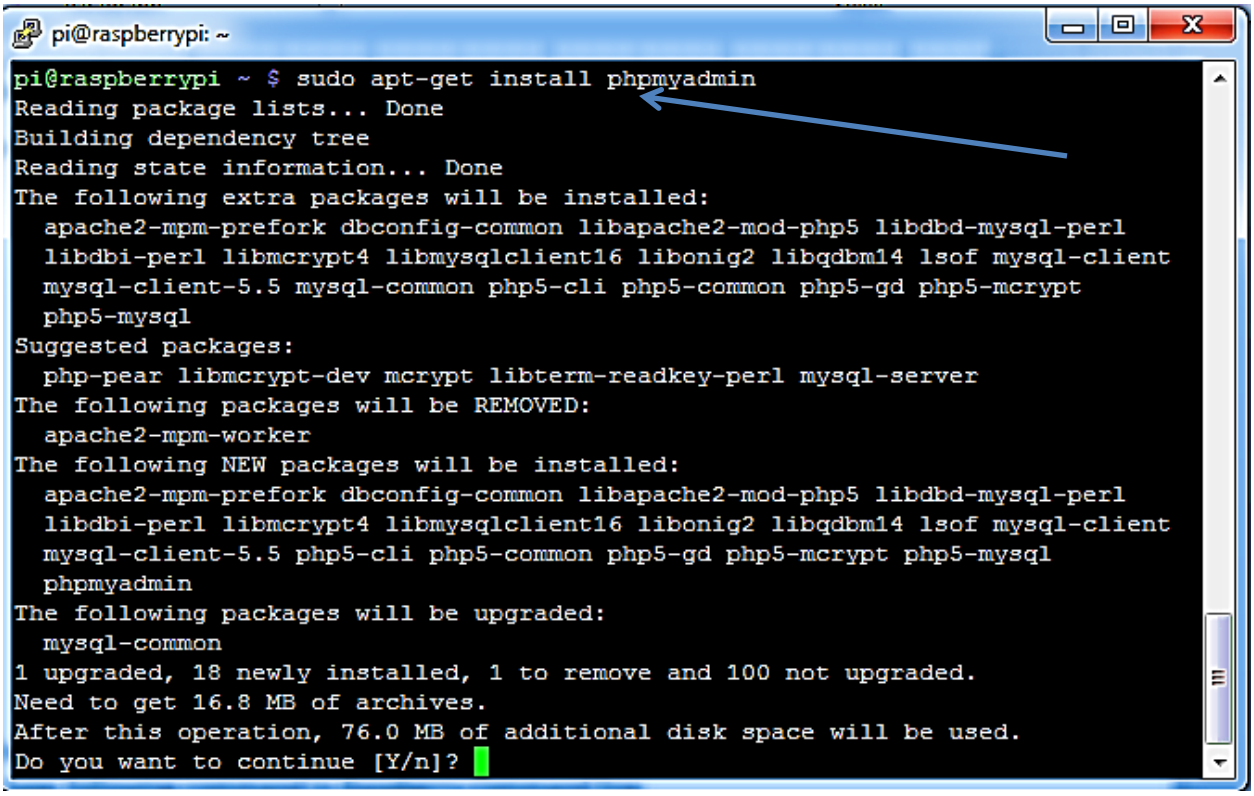
Sudo apt-get install apache2 –y

After successful installation type ip address in web page it will look something like this



Step2: installing phpmyadmin
To install php type following command in Raspberry command line

sudo apt-get install phpmyadmin

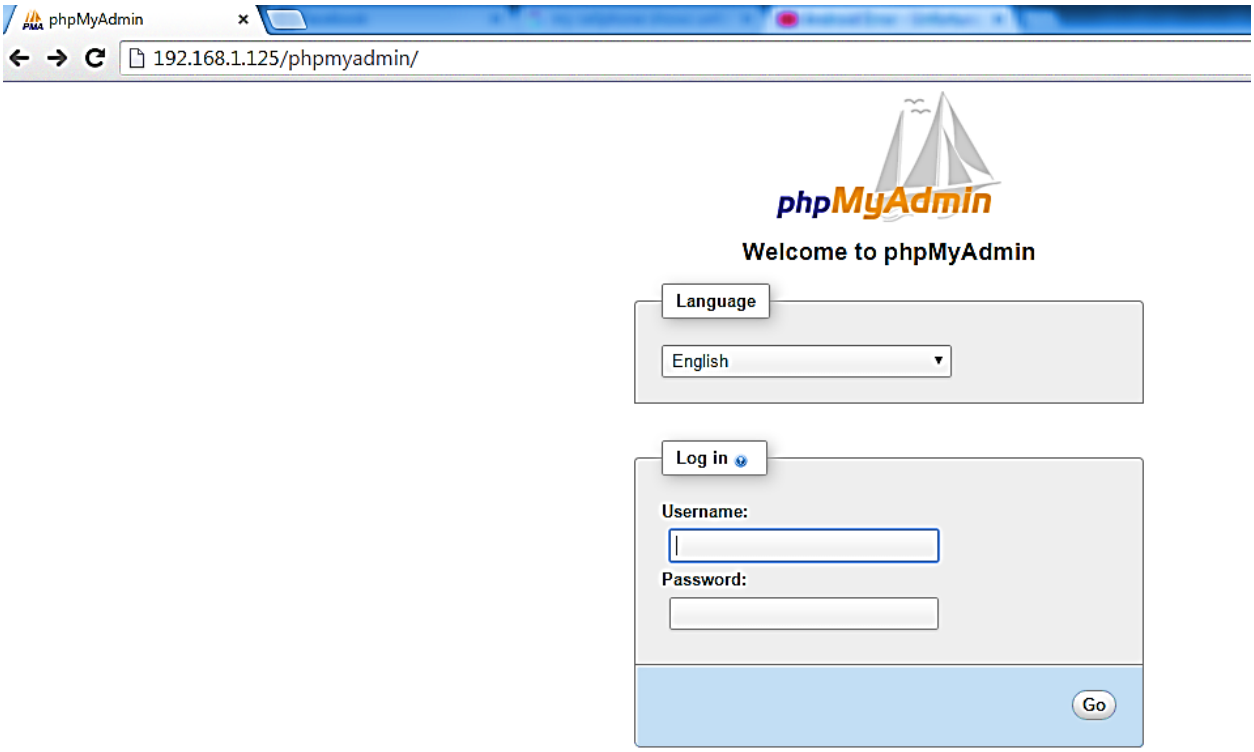


Step3:
configure apache to work with phpmyadmin
type **sudo nano /etc/apache2/apache2.conf**
add the line **include /etc/phpmyadmin/apache.conf** at the end of the line

Step4: restart apache by following command
sudo /etc/init.d/apache2 restart

```
pi@raspberrypi: ~
pi@raspberrypi ~ $ sudo nano /etc/apache2/apache2.config
pi@raspberrypi ~ $ sudo nano /etc/apache2/apache2.confi
pi@raspberrypi ~ $ sudo nano /etc/apache2/apache2.conf
pi@raspberrypi ~ $ /etc/init.d/apache2 restart
[....] Restarting web server: apache2/usr/sbin/apache2ctl: 87: ulimit: error set
ting limit (Operation not permitted)
apache2: Could not reliably determine the server's fully qualified domain name,
using 127.0.1.1 for ServerName
/usr/sbin/apache2ctl: 87: ulimit: error setting limit (Operation not permitted)
apache2: Could not reliably determine the server's fully qualified domain name,
using 127.0.1.1 for ServerName
(13)Permission denied: make_sock: could not bind to address 0.0.0.0:80
no listening sockets available, shutting down
Unable to open logs
Action 'start' failed.
The Apache error log may have more information.
 failed!
pi@raspberrypi ~ $ sudo /etc/init.d/apache2 restart
[....] Restarting web server: apache2apache2: Could not reliably determine the s
erver's fully qualified domain name, using 127.0.1.1 for ServerName
 ... waiting apache2: Could not reliably determine the server's fully qualified
domain name, using 127.0.1.1 for ServerName
. ok
pi@raspberrypi ~ $
```

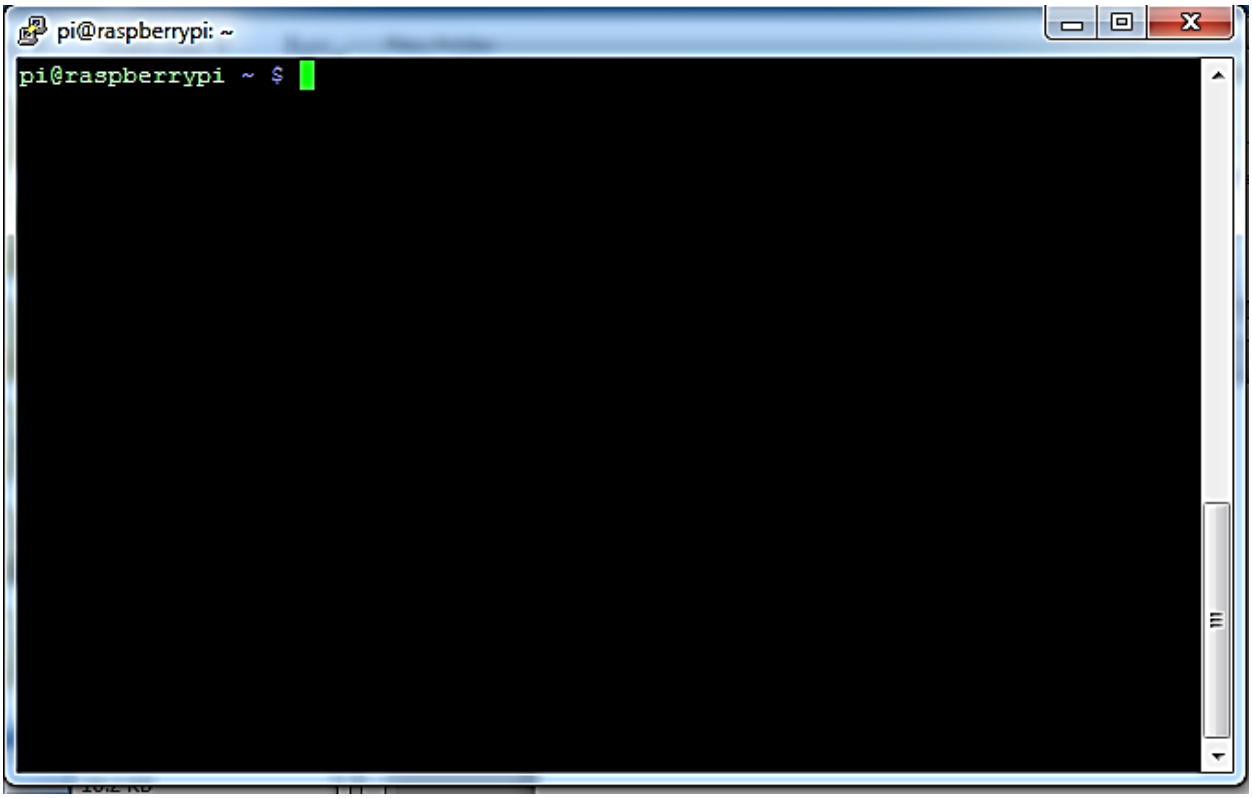
Step5: now if we type ipaddress/phpmyadmin in web page screen will look something like this



WORKING WITH RFID USING RASPBERRY PI:

Components:
Raspberry pi
USB RFID read
Can buy from: http://researchdesignlab.com/usb-rfid-reader.html

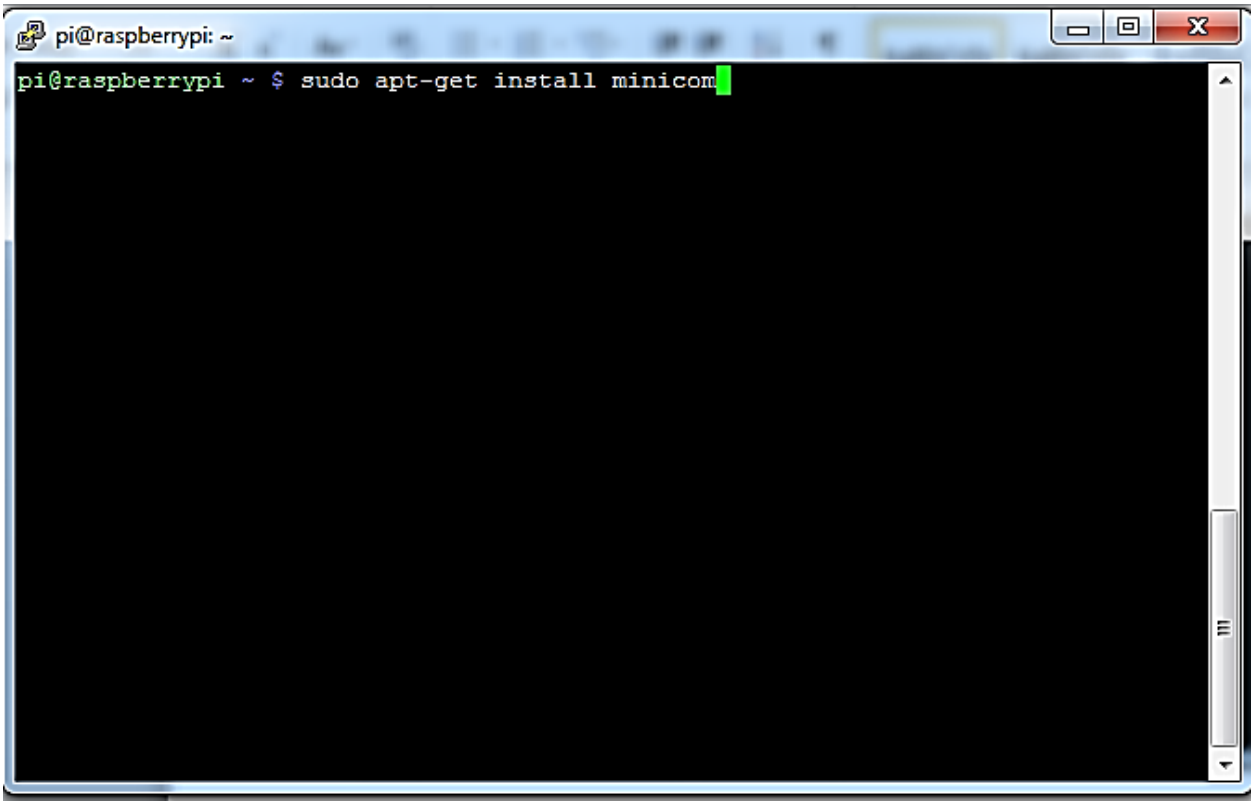
Step1: login Rpi



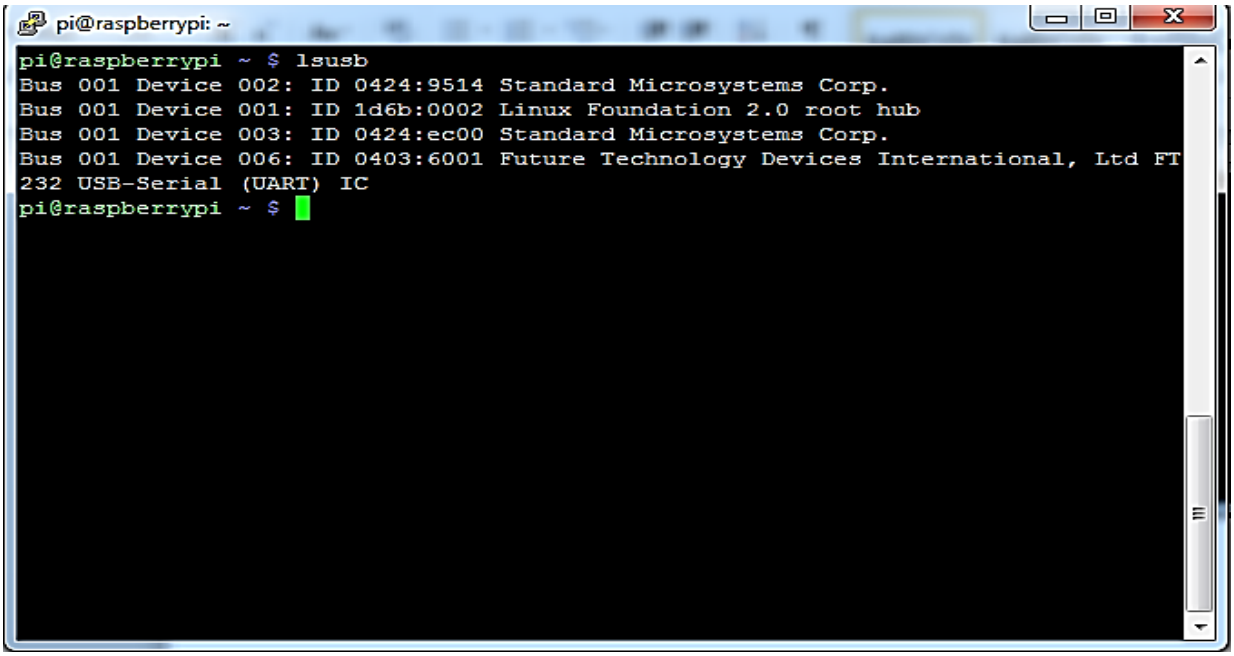
Step2: connect RFID to USB0 port of raspberry i.e first usb port

Step3: install minicom by typing following command

sudo apt-get install minicom

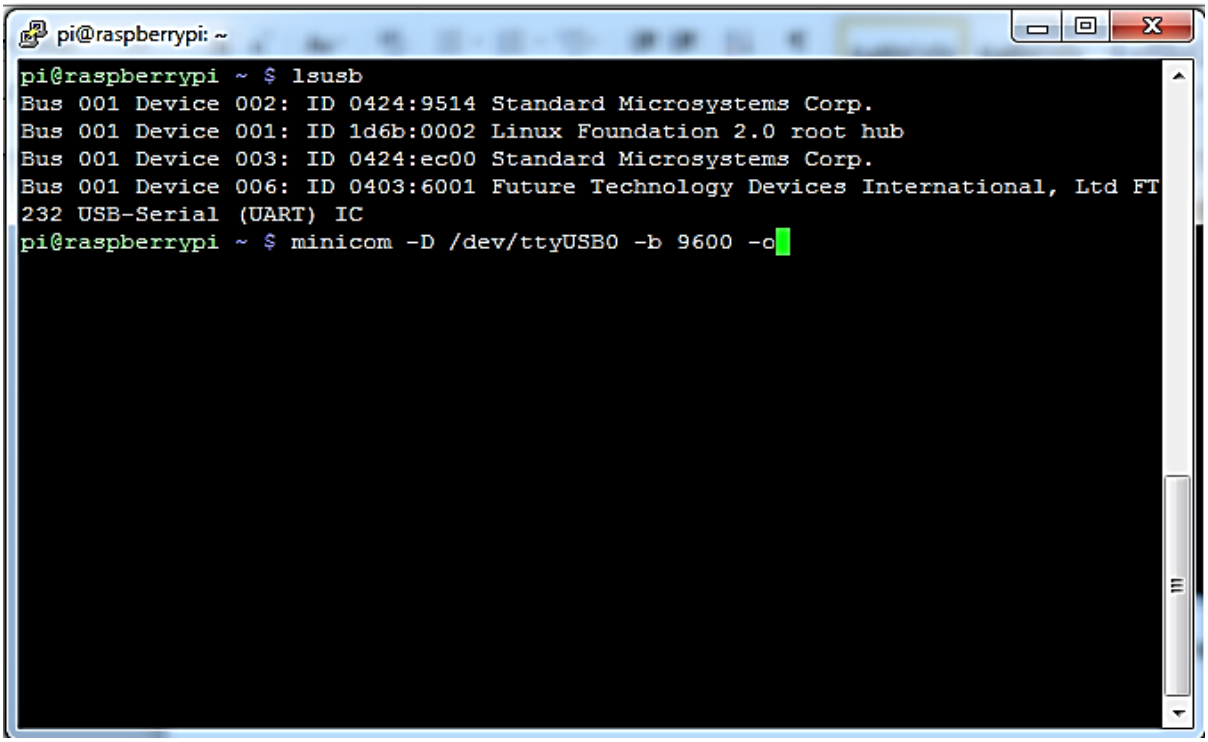


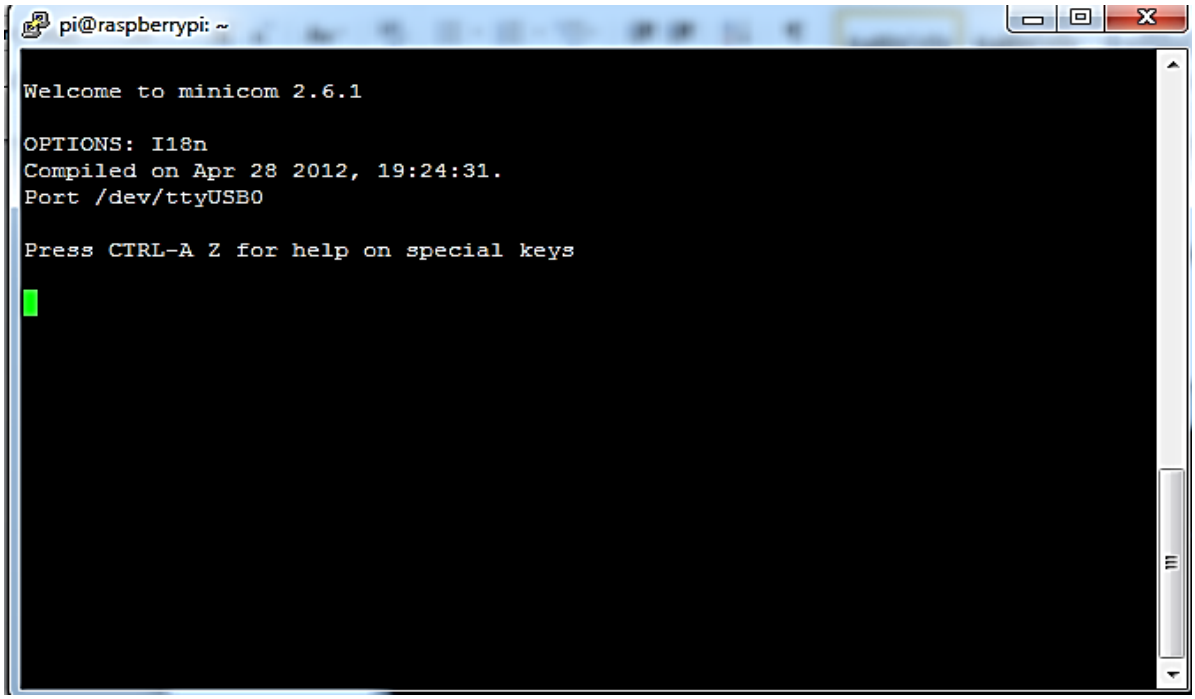
Step 4: Check whether RFID is connected to USB by using **lsusb** command



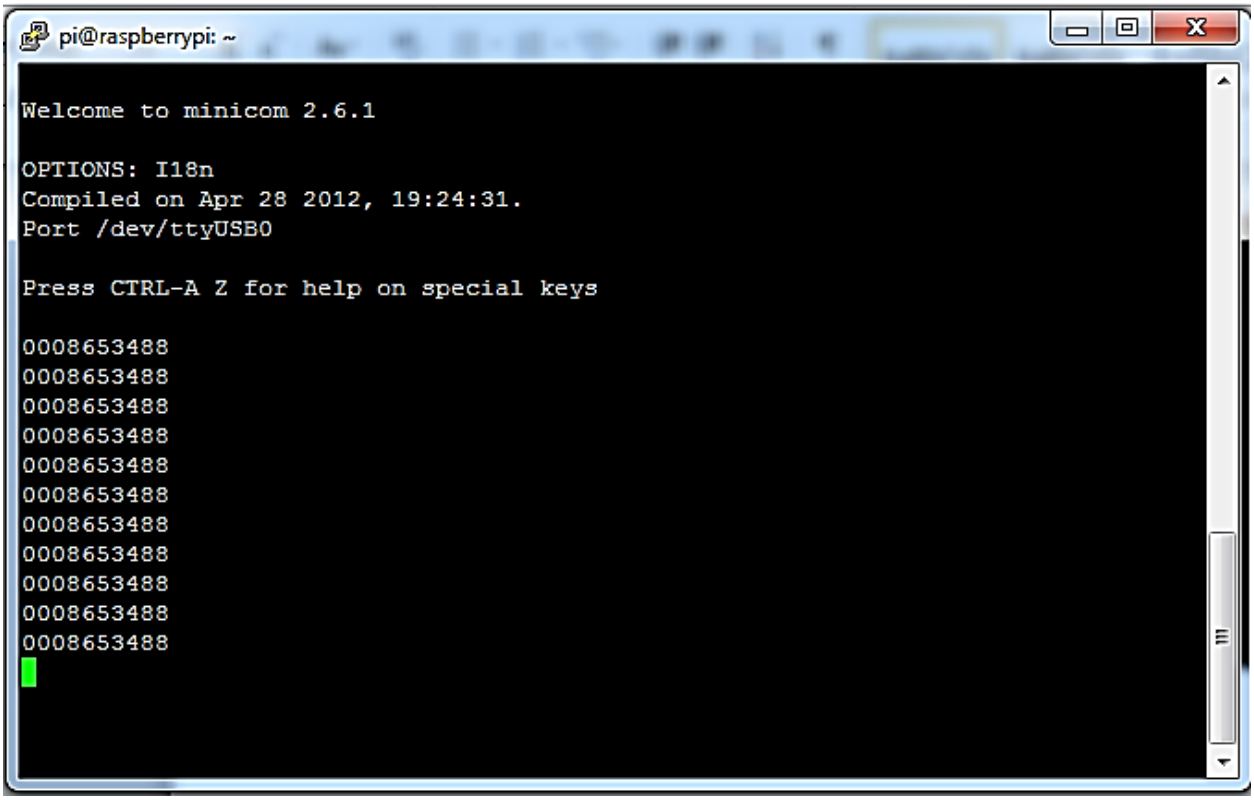
Step5: get minicom window by setting baud rate of 9600 and specifying device path path by typing following command

minicom –D /dev/ttyUSB0 –b 9600 –o

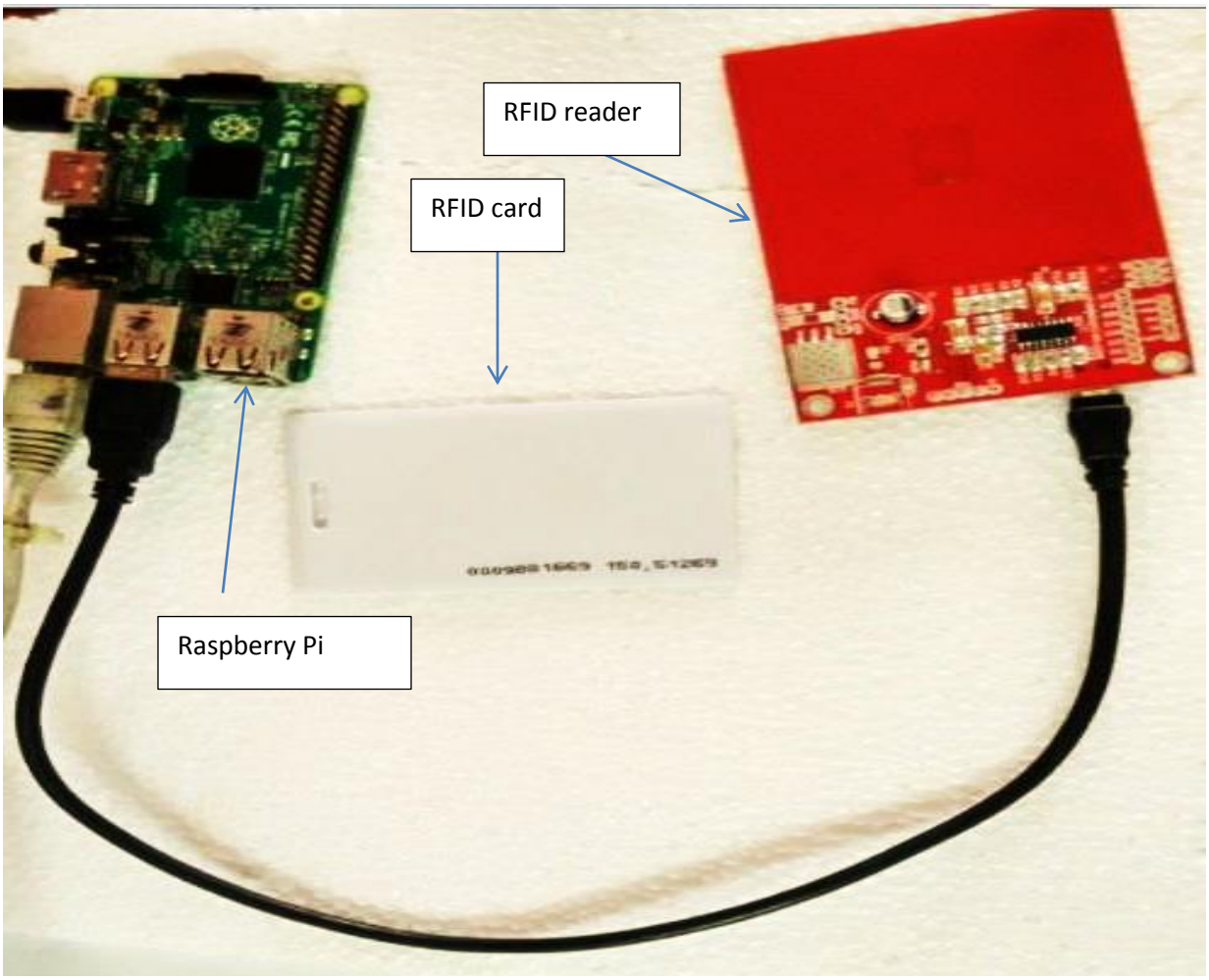




Step6: Finally move RFID card over reader you can see card information displaying on minicom serial window



Connection:


RFID reader
RFID card
Raspberry Pi

SERIAL COMMUNICATION:

1. INTERFACING GSM AND RASPBERRY PI:

Components:
Raspberry Pi
Can buy from: http://researchdesignlab.com/development-baord/raspberry-pi-40/raspberry-pi-model-b.html
GSM SIM900A modem
Can buy from: http://researchdesignlab.com/gsm-sim-900.html
USB to TTL converter
Can buy from: http://researchdesignlab.com/index.php/modules/usb-to-rs232-converter.html
Female jumper wires
Can buy from: http://researchdesignlab.com/jumper-wire-female-pack-of-5.html

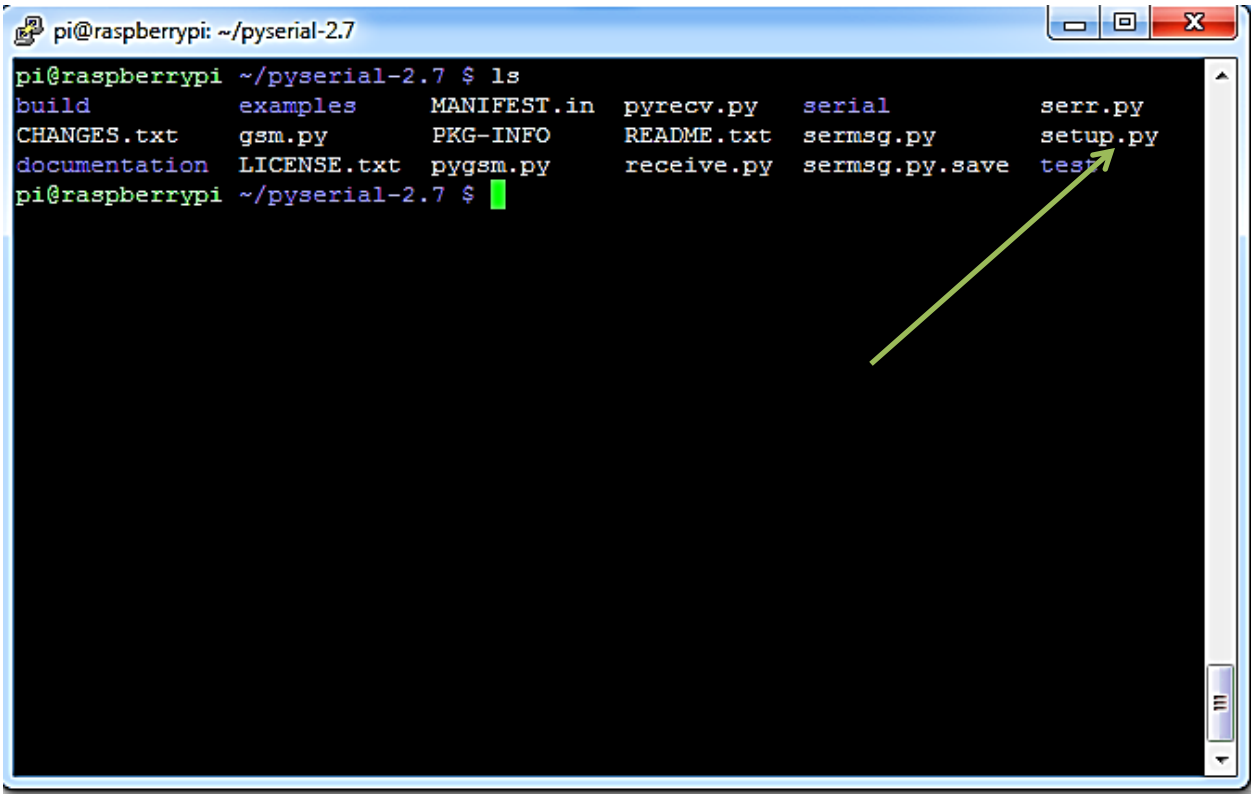
The following step explains how to communicate external devices serially using serial communication
Step1: initially we have to download and install python serial package called py serial from https://pypi.python.org/pypi/pyserial

Step2: unzip the file and enter into that folder using command

cd "package name"

Step3: inside that there will be setup file known as setup.py. install software by running command

sudo python setup.py install



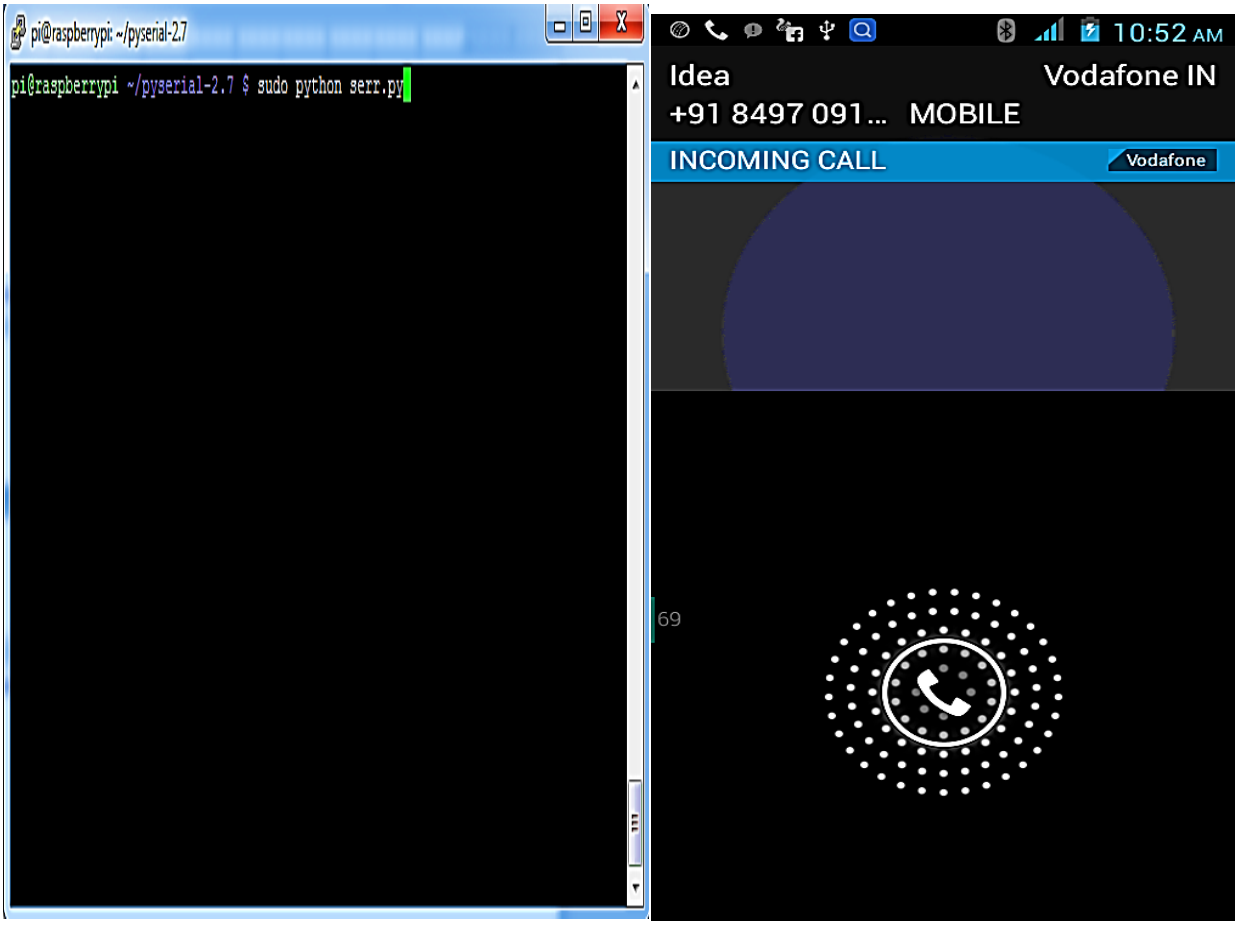
Program:

Case 1: making call from Raspberry pi using GSM modem

```
import serial                                          (1)
import time                                            (2)
ser  = serial.Serial('/dev/ttyUSB0',9600,timeout=1)    (3)
ser.flush()
ser.write('ATD888xxxxxxx;\r')                          (4)
time.sleep(10)                                         (5)
ser.close()                                            (6)
```

1) This command is used to call serial package which is installed before
2) This command is for delay function
3) Indicates to which USB port device is connected
4) This command is used to make a call from GSM module to specified number
5) It is used to give a delay
6) Closes the serial communication

Run a code by **sudo python 'filename.py'**

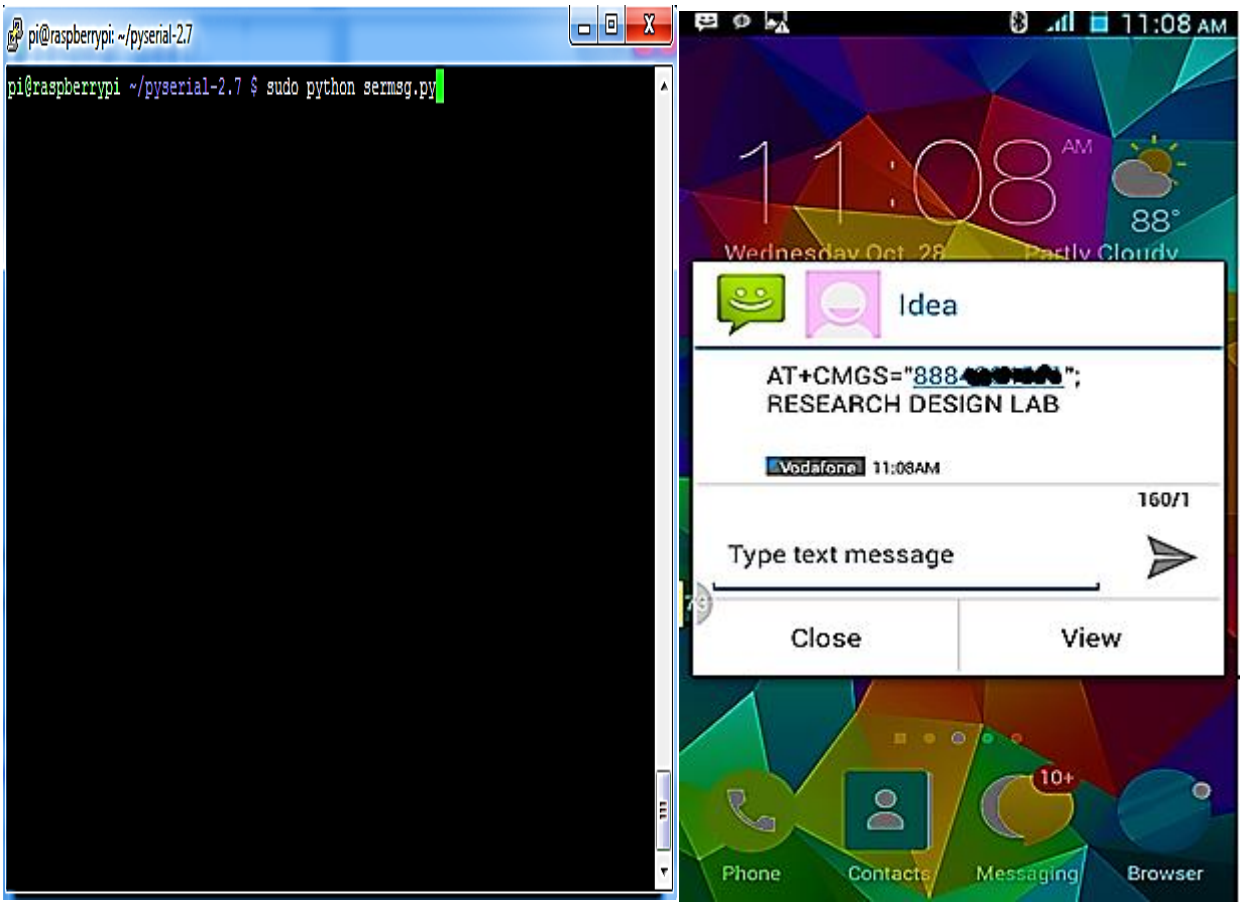


Case2:
Sending message from Raspberry using GSM modem

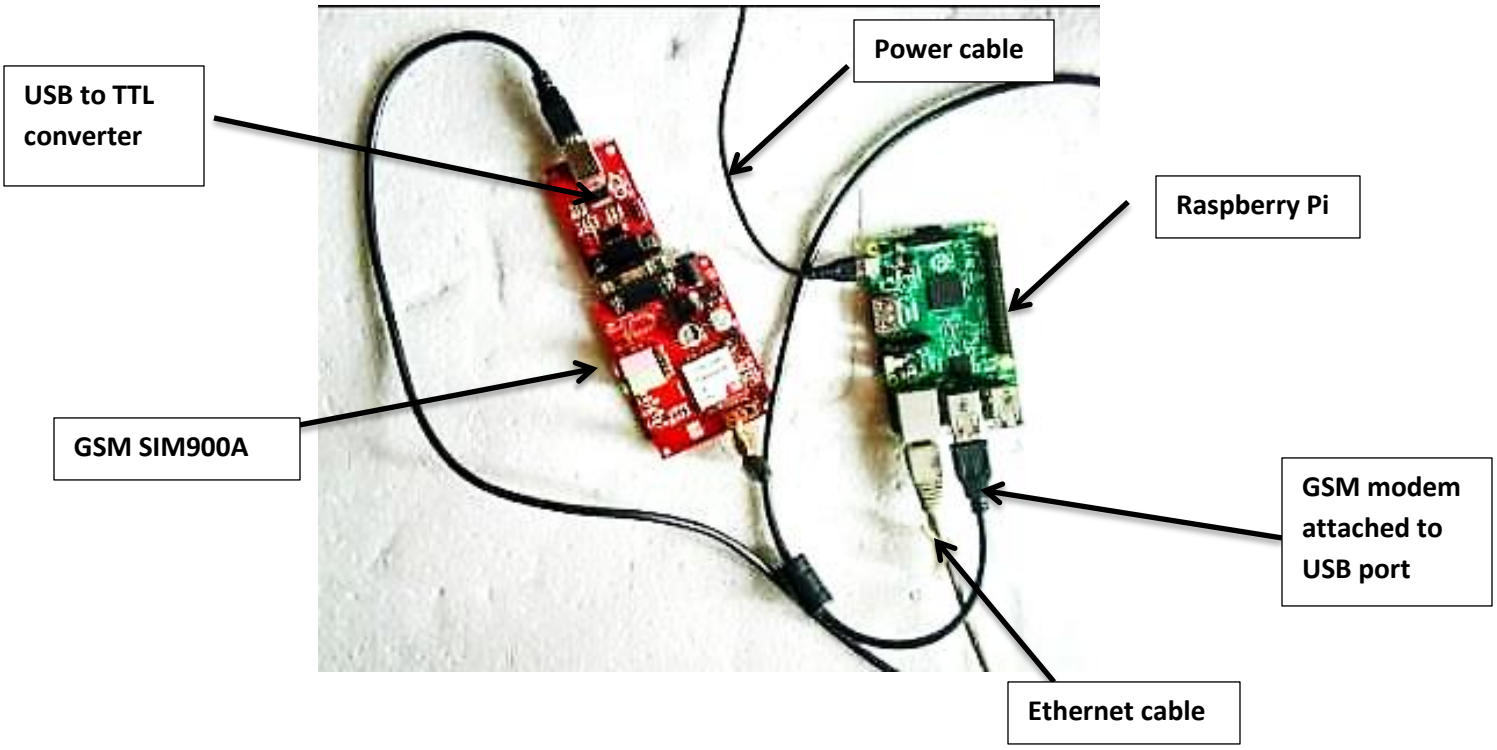
```
import serial                                          (1)
import time(2)
ser  = serial.Serial('/dev/ttyUSB0',9600,timeout=1)    (3)
ser.flush()
ser.write('AT+CMGS="888xxxxxxx";\r\n')
ser.write('ddddd'+'\r\n')                              (4)
ser.write('\x1A')
time.sleep(10)                                         (5)
ser.close()                                            (6)
```

1) This command is used to call serial package which is installed before
2) This command is for delay function
3) Indicates to which USB port device is connected
4) This command is used to send a message from GSM module to specified number
5) It is used to give a delay
6) Closes the serial communication

Run a code by **sudo python 'filename.py'**



Connections:



2. CONNECTING ARDUINO TO RASPBERRY PI:

Step1: firstly we have to install python serial package. Follow the same procedure as mentioned in previous experiment

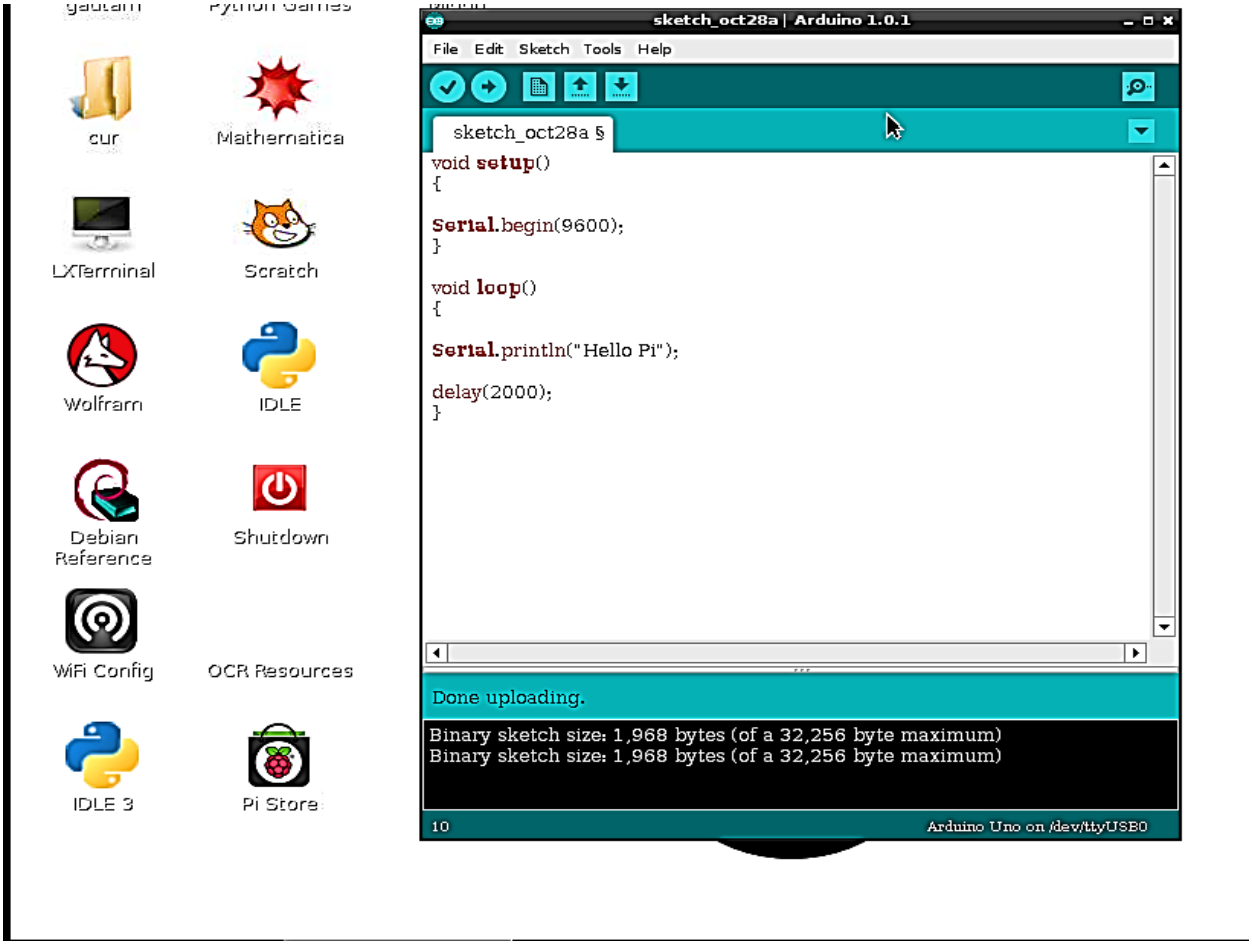
Step2:
Next we have to install arduino raspberry, which can be done by using command

Sudo apt-get install arduino

Step3: open a sketch and type following program in it

```
void setup(){
Serial.begin(9600);
}

void loop(){
Serial.println("Hello Pi");
delay(2000);
}
```

This program prints hello pi in raspberry serial terminal.



Step4: install python package pyserial as mentioned in previous examples

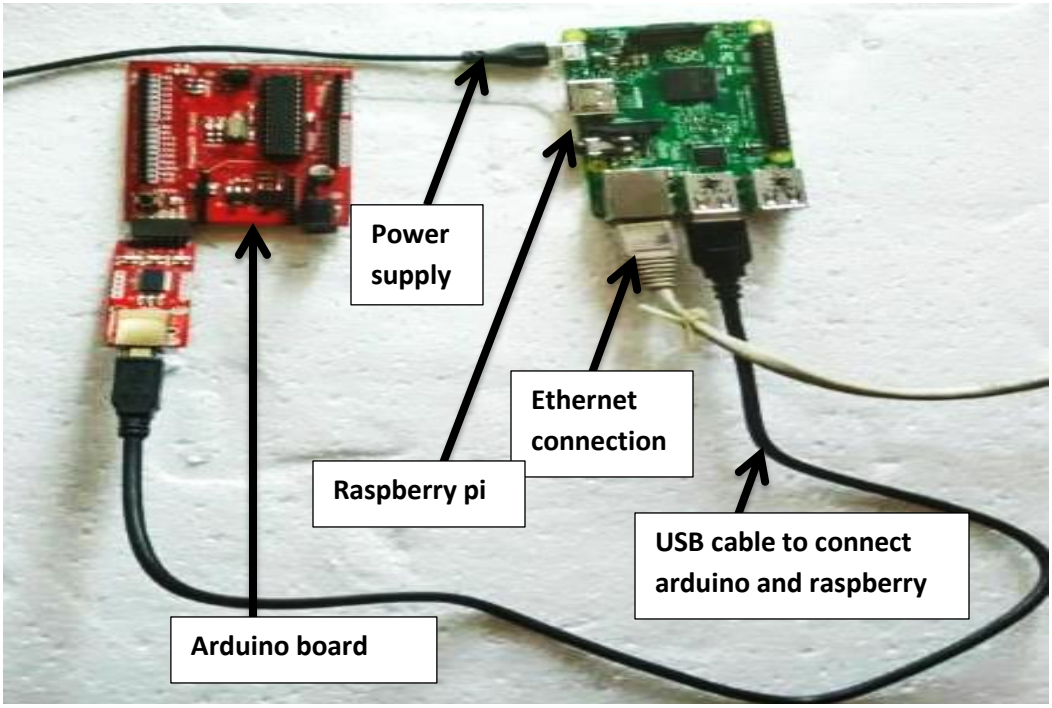
Step5: in raspberry pi run **sudo python** and in command line type following commands

```
import serial

ser = serial.Serial('/dev/ttyACM0', 9600)

while 1:

        ser.readline()
```

this will print Hello Pi message from arduino

```
pi@raspberrypi: ~/pyserial-2.7
pi@raspberrypi ~/pyserial-2.7 $ sudo python
Python 2.7.3 (default, Mar 18 2014, 05:13:23)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import serial
>>> ser = serial.Serial('/dev/ttyUSB0',9600)
>>> while 1:
...     ser.readline()
...
'Hello Pi\r\n'
'Hello Pi\r\n'
'Hello Pi\r\n'
'Hello Pi\r\n'
'Hello Pi\r\n'
'Hello Pi\r\n'
'Hello Pi\r\n'
'Hello Pi\r\n'
'Hello Pi\r\n'
'Hello Pi\r\n'
```

connections:

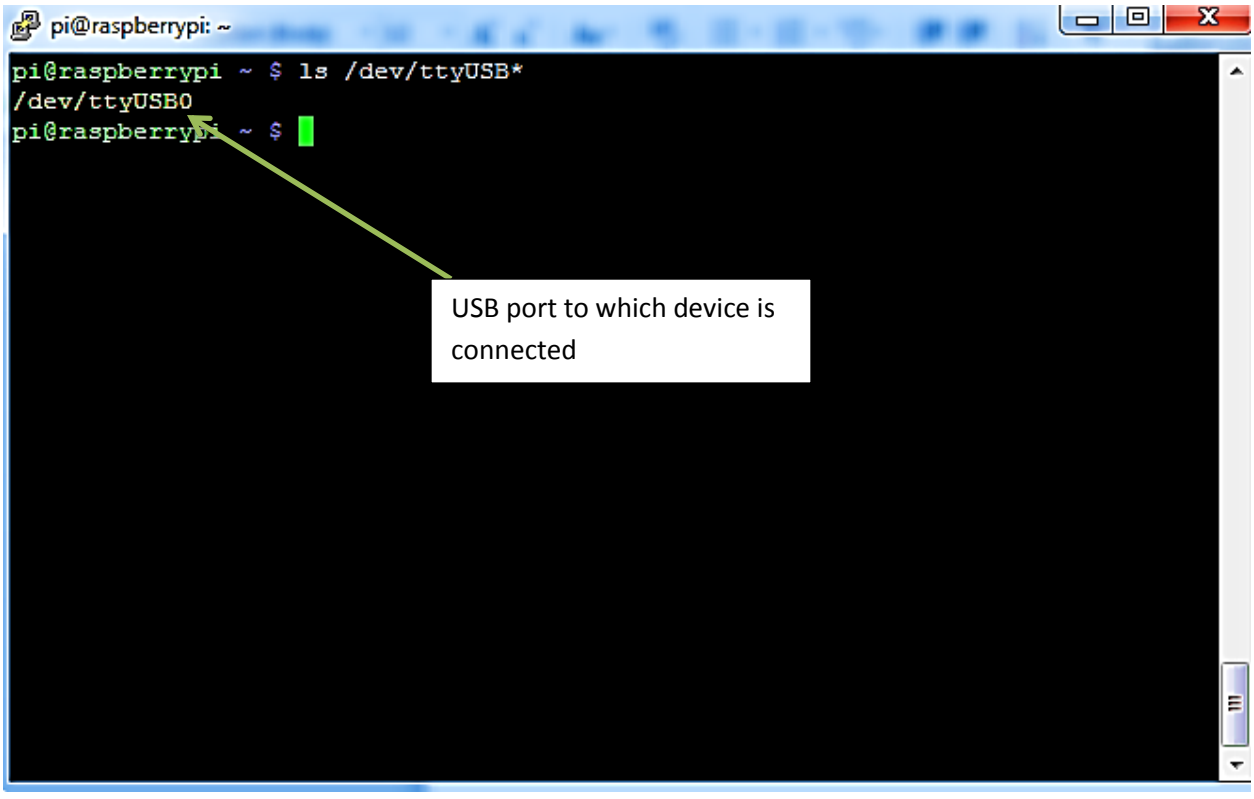


3. CONNECTING GPS TO RASPBERRY PI:

Components :
Raspberry Pi
Can buy from: http://researchdesignlab.com/development-baord/raspberry-pi-40/raspberry-pi-model-b.html
GPS Receiver PA6E-CAM with GPS Antenna
Can buy from: http://researchdesignlab.com/gps-receiver-pa6e-cam-with-gps-antenna.html
USB to TTL converter
Can buy from: http://researchdesignlab.com/index.php/modules/usb-to-rs232-converter.html
Female jumper wires
Can buy from: http://researchdesignlab.com/jumper-wire-female-pack-of-5.html

Step1:
Connect GPS system to USB to TTL and to USB port of an Raspberry Pi. Use following command to check to which USB port it is connected it shows respective USB port

ls /dev/ttyUSB*



Step2: installing required software's. install gpsd software by using following command

sudo apt-get install gpsd gpsd-clients python-gps

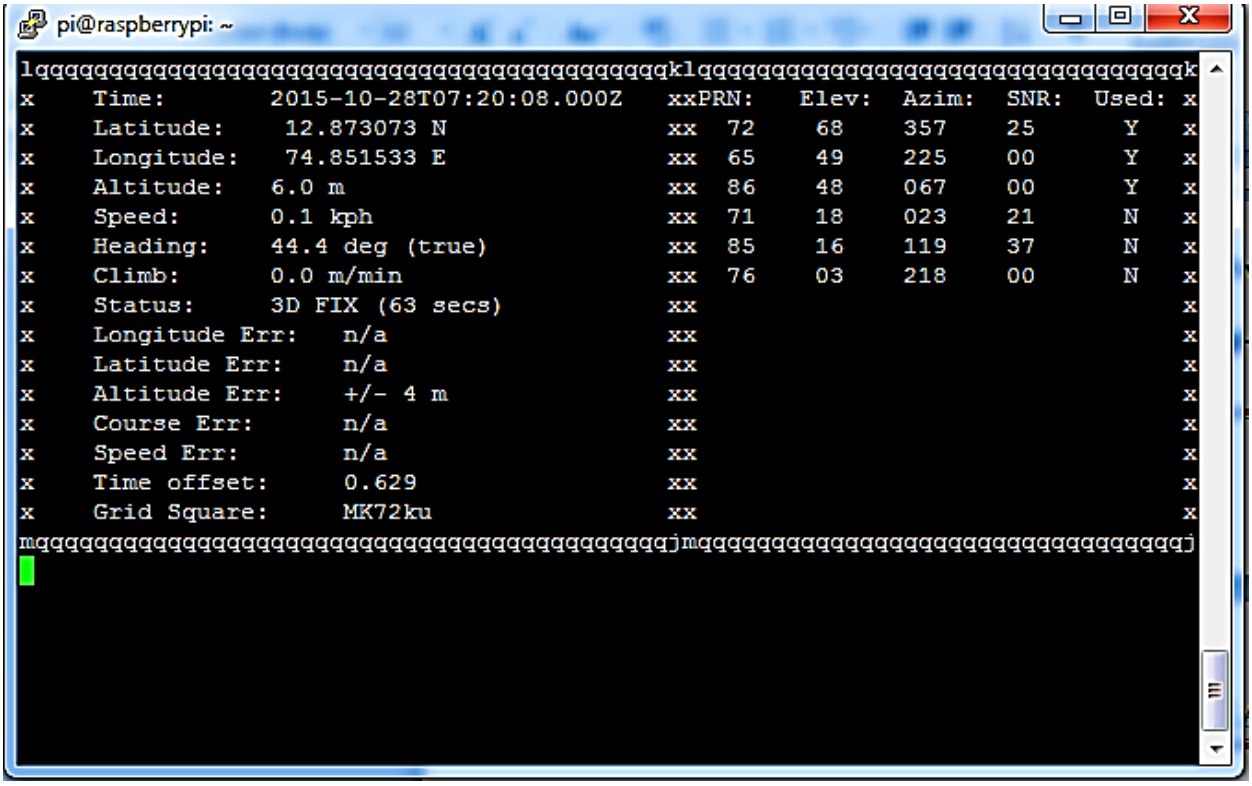
Step3: use below command to point out the our GPS device on the USB to TTL adapter

sudo gpsd /dev/ttyUSB0 -F /var/run/gpsd.sock

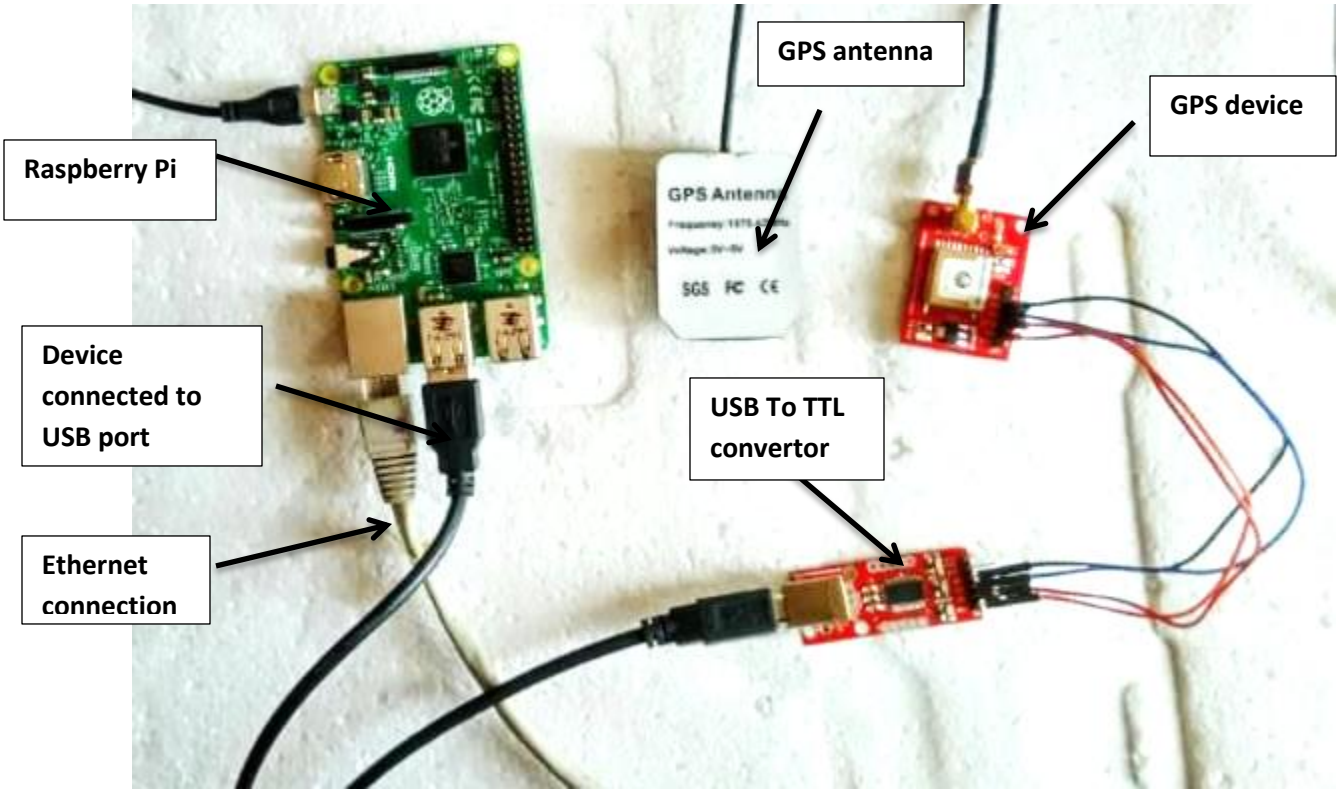
Step4: type the following command to test the result

cgps -s

it will show a output similar to this on the screen with all GPS information



Connections:


GPS antenna
GPS device
Raspberry Pi
Device connected to USB port
USB To TTL convertor
Ethernet connection