

Practical 1:

Aim: LINUX COMMANDS: EXPLORING THE RASPBIAN

LINUX COMMANDS

Here are some fundamental and common Linux commands with example usage:

FILESYSTEM commands:

- **LS** : The ls command lists the content of the current directory (or one that is specified). It can be used with the -l flag to display additional information (permissions, owner, group, size, date and timestamp of last edit) about each file and directory in a list format. The -a flag allows you to view files beginning with . (i.e. dotfiles).
- **CD** : Using cd changes the current directory to the one specified. You can use relative (i.e. cd directoryA) or absolute (i.e. cd /home/pi/directoryA) paths.
- **PWD** : The pwd command displays the name of the present working directory: on a Raspberry Pi, entering pwd will output something like /home/pi.
- **MKDIR** : You can use mkdir to create a new directory, e.g. mkdir newDir would create the directory newDir in the present working directory.
- **RMDIR** : To remove empty directories, use rmdir. So, for example, rmdir oldDir will remove the directory oldDir only if it is empty.
- **RM** : The command rm removes the specified file (or recursively from a directory when used with -r). Be careful with this command: files deleted in this way are mostly gone for good!
- **CP** : Using cp makes a copy of a file and places it at the specified location (this is similar to copying and pasting). For example, cp ~/fileA /home/otherUser/ would copy the file fileA from your home directory to that of the user otherUser (assuming you have permission to copy it there). This command can either take FILE FILE (cp fileA fileB), FILE

DIR(cp fileA /directoryB/) or -r DIR DIR (which recursively copies the contents of directories) as arguments.

- **MV** : The mv command moves a file and places it at the specified location (so where cp performs a 'copy-paste', mv performs a 'cut-paste'). The usage is similar to cp. So mv ~/fileA /home/otherUser/ would move the file fileA from your home directory to that of the user otherUser. This command can either take FILE FILE (mv fileA fileB), FILE DIR (mv fileA /directoryB/) or DIR DIR (mv /directoryB /directoryC) as arguments. This command is also useful as a method to rename files and directories after they've been created.
- **TOUCH** : The command touch sets the last modified time-stamp of the specified file(s) or creates it if it does not already exist.
- **CAT** : You can use cat to list the contents of file(s), e.g. cat thisFile will display the contents of thisFile. Can be used to list the contents of multiple files, i.e. cat *.txt will list the contents of all .txt files in the current directory.
- **HEAD** : The head command displays the beginning of a file. Can be used with -n to specify the number of lines to show (by default ten), or with -c to specify the number of bytes.
- **TAIL** : The opposite of head, tail displays the end of a file. The starting point in the file can be specified either through -b for 512 byte blocks, -c for bytes, or -n for number of lines.
- **CHMOD** : You would normally use chmod to change the permissions for a file. The chmod command can use symbols u (user that owns the file), g (the files group) , and o (other users) and the permissions r (read), w (write), and x (execute). Using chmod u+x *filename* will add execute permission for the owner of the file.
- **CHOWN** : The chown command changes the user and/or group that owns a file. It normally needs to be run as root using sudo e.g. sudo chown pi:root *filename* will change the owner to pi and the group to root.
- **SSH** : ssh denotes the secure shell. Connect to another computer using an encrypted network connection. For more details see SSH (secure shell)

- **SCP** : The scp command copies a file from one computer to another using ssh. For more details see SCP (secure copy)
- **SUDO** : The sudo command enables you to run a command as a superuser, or another user. Use sudo -s for a superuser shell. For more details see Root user / sudo
- **DD** : The dd command copies a file converting the file as specified. It is often used to copy an entire disk to a single file or back again. So, for example, dd if=/dev/sdd of=backup.img will create a backup image from an SD card or USB disk drive at /dev/sdd. Make sure to use the correct drive when copying an image to the SD card as it can overwrite the entire disk.
- **DF** : Use df to display the disk space available and used on the mounted filesystems. Use df -h to see the output in a human-readable format using M for MBs rather than showing number of bytes.
- **UNZIP** : The unzip command extracts the files from a compressed zip file.
- **TAR** : Use tar to store or extract files from a tape archive file. It can also reduce the space required by compressing the file similar to a zip file. To create a compressed file, use tar -cvzf *filename.tar.gz* *directory/* To extract the contents of a file, use tar -xvzf *filename.tar.gz*
- **ssinput** for another command. The pipe symbol is a vertical line |. For example, to only show the first ten entries of the ls command it can be piped through the head command ls | head
- **TREE** : Use the tree command to show a directory and all subdirectories and files indented as a tree structure. & Run a command in the background with &, freeing up the shell for future commands.
- **WGET** : Download a file from the web directly to the computer with wget. So wget <https://www.raspberrypi.org/documentation/linux/usage/commands.md> will download this file to your computer as commands.md
- **CURL** : Use curl to download or upload a file to/from a server. By default, it will output the file contents of the file to the screen.

- **MAN** : Show the manual page for a file with man. To find out more, run `man man` to view the manual page of the man command.

SEARCH COMMANDS

- **GREP** : Use grep to search inside files for certain search patterns. For example, `grep "search" *.txt` will look in all the files in the current directory ending with .txt for the string search. The find command searches a directory and subdirectories for files matching certain patterns.
- **WHEREIS** : Use whereis to find the location of a command. It looks through standard program locations until it finds the requested command.

NETWORKING COMMANDS:

- **PING** : The ping utility is usually used to check if communication can be made with another host. It can be used with default settings by just specifying a hostname (e.g. `ping raspberrypi.org`) or an IP address (e.g. `ping 8.8.8.8`). It can specify the number of packets to send with the `-c` flag.
- **NMAP** : nmap is a network exploration and scanning tool. It can return port and OS information about a host or a range of hosts. Running just `nmap` will display the options available as well as example usage.
- **HOSTNAME** : The hostname command displays the current hostname of the system. A privileged (super) user can set the hostname to a new one by supplying it as an argument (e.g. `hostname new-host`).
- **IFCONFIG** : Use ifconfig to display the network configuration details for the interfaces on the current system when run without any arguments (i.e. `ifconfig`). By supplying the command with the name of an interface (e.g. `eth0` or `lo`) you can then alter the configuration.

Practical 2:

Aim : To Perform Bash Commands

1. Create a bash file with following structure
f1/f2/t1

GNU nano 2.0.6

File: file1

```
#!/bin/bash
```

```
mkdir f1
```

```
cd f1
```

```
mkdir f2
```

```
cd f2
```

```
touch t1
```

```
Rishabh:~ rishabhghodke$ ./file1
```

```
Rishabh:~ rishabhghodke$ chmod +x file1
```

```
Rishabh:~ rishabhghodke$ ls
```

#.profile#	Library	addition
AndroidStudioProjects	Movies	enthought
Applications	Music	f1
Desktop	Pictures	file1
Documents	Projects	iCloud Drive (Archive)
Downloads	Public	s.py

2.Create a bash file with following

- a. man pwd
- b. man touch
- c. man cp

```
[Rishabh:~ rishabhghodke$ nano file2
[Rishabh:~ rishabhghodke$ chmod + file2
[Rishabh:~ rishabhghodke$ chmod +x file2
[Rishabh:~ rishabhghodke$ ./file2

[1]+  Stopped                  ./file2
Rishabh:~ rishabhghodke$
```

PWD(1)	BSD General Commands Manual	PWD(1)
--------	-----------------------------	--------

NAME

`pwd` -- return working directory name

SYNOPSIS

`pwd [-L | -P]`

DESCRIPTION

The `pwd` utility writes the absolute pathname of the current working directory to the standard output.

3.Create a bash file to perform Arithmetic Operations

```
Rishabh:~ rishabhghodke$ nano file3
Rishabh:~ rishabhghodke$ chmod +x file3
Rishabh:~ rishabhghodke$ ./file3
Rishabh:~ rishabhghodke$ cd f1
Rishabh:f1 rishabhghodke$ ls
f2      t1
Rishabh:f1 rishabhghodke$
```

GNU nano 2.0.6

File: file4

```
#!/bin/bash
echo "arithmetic operation"
echo "First number"
read a
echo "Second number"
read b
c=`expr $a + $b`
d=$((a-b))
echo "Addition is $c"
echo "Subtraction is $d"
```



```
[Rishabh:fl rishabhghodke$ ./file4
arithmetic operation
First number
4
Second number
8
Addition is 12
Subtraction is -4
Rishabh:fl rishabhghodke$
```

4. Create a bash file to check whether numbers are same

```
#!/bin/bash
echo "arithmetic operation"
echo "First number"
read a
echo "Second number"
read b
if [ $a -eq $b ]
then
echo "same number"
else
echo "different number"
fi
```

```
Rishabh:fl rishabhghodke$ ./file5
arithmetic operation
First number
6
Second number
9
different number
Rishabh:fl rishabhghodke$ ./file5
arithmetic operation
First number
6
Second number
6
same number
Rishabh:fl rishabhghodke$ █
```

5.Create a bash file to perform Multiplication

```
#!/bin/bash
echo "arithmetic operation"
echo "First number"
read a
echo "Second number"
read b
c= `expr "$a" \* "$b"`
echo "multiplication is $c"
█
```

```
Rishabh:fl rishabhghodke$ ./file6
arithmetic operation
First number
8
Second number
5
multiplication is 40
Rishabh:fl rishabhghodke$
```

6.Create a bash file to perform switch case operation

```
#!/bin/bash
echo "switch case"
echo "choose any randon number from 1 to 5"
read no;
case $no in
    1)echo "good boy" ;;
    2)echo "hello World" ;;
    3)echo "hi" ;;
    4)echo "bye" ;;
    5)echo "talk to you later" ;;
    *)echo "good morning" ;;
esac
```

```
Rishabh:fl rishabhghodke$ ./file7  
switch case  
choose any randon number from 1 to 5  
5  
talk to you later  
Rishabh:fl rishabhghodke$
```

2.Create Bash File to create Touch file m1,m2,m3,m4

```
#!/bin/bash  
touch m1  
touch m2  
touch m3  
touch m4
```

```
Rishabh:f1 rishabhghodke$ chmod +x file9
Rishabh:f1 rishabhghodke$ ./file9
Rishabh:f1 rishabhghodke$ ls
f2      file5   file7   file9   m2      m4
file4   file6   file8   m1      m3      t1
Rishabh:f1 rishabhghodke$
```

3.Create Bash File to create cat file m1,m2,m3,m4

```
#!/bin/bash
```

```
cat m1
```

```
cat m2
```

```
cat m3
```

```
cat m4
```

```
Rishabh:f1 rishabhghodke$ chmod +x file10
```

```
Rishabh:f1 rishabhghodke$ ./file10
```

```
Rishabh:f1 rishabhghodke$ ls
```

```
f2      file4    file6    file8    m1        m3        t1
```

```
file10  file5    file7    file9    m2        m4
```

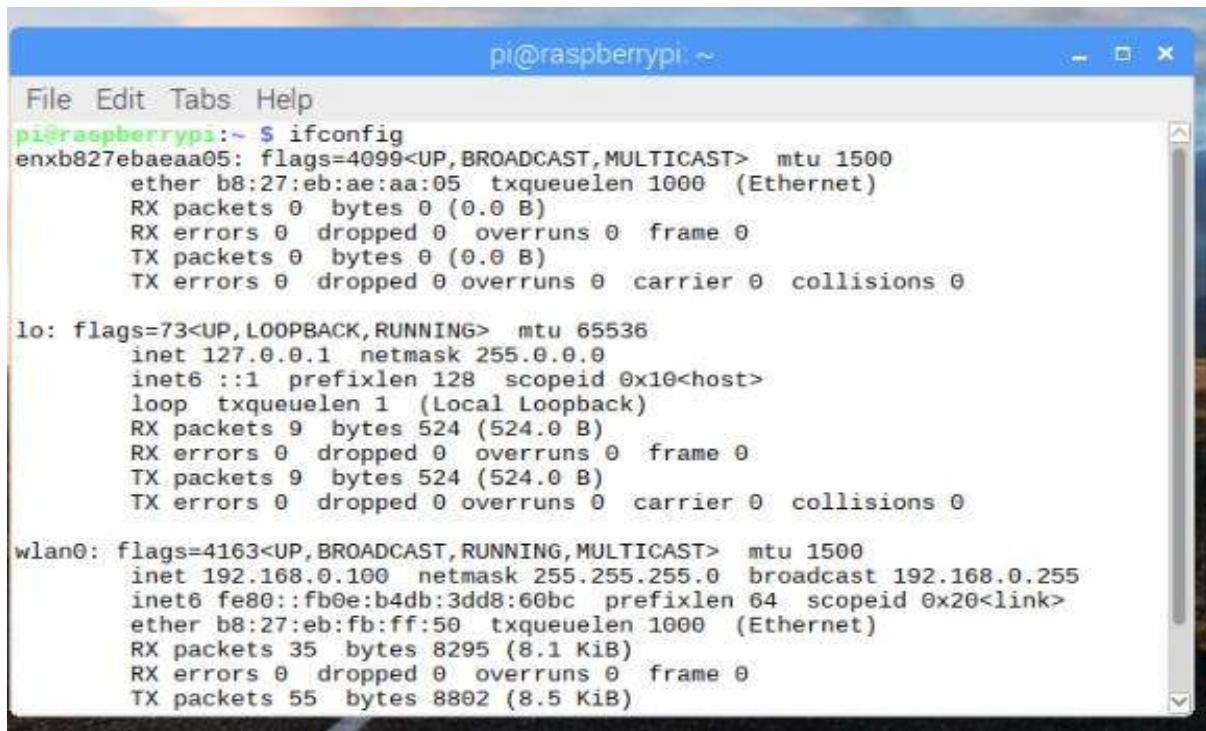
```
Rishabh:f1 rishabhghodke$
```

Practical 3:

Aim : Learn to ssh into a raspberry pi using putty from a different computer on the network.

Steps for transferring file from windows to raspberry:

Step 1: Find out the IP address of raspberry - ifconfig



```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~ $ ifconfig  
enxb827ebaeaa05: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500  
    ether b8:27:eb:ae:aa:05 txqueuelen 1000 (Ethernet)  
    RX packets 0 bytes 0 (0.0 B)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 0 bytes 0 (0.0 B)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536  
    inet 127.0.0.1 netmask 255.0.0.0  
    inet6 ::1 prefixlen 128 scopeid 0x10<host>  
    loop txqueuelen 1 (Local Loopback)  
    RX packets 9 bytes 524 (524.0 B)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 9 bytes 524 (524.0 B)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
    inet 192.168.0.100 netmask 255.255.255.0 broadcast 192.168.0.255  
    inet6 fe80::fb0e:b4db:3dd8:60bc prefixlen 64 scopeid 0x20<link>  
    ether b8:27:eb:fb:ff:50 txqueuelen 1000 (Ethernet)  
    RX packets 35 bytes 8295 (8.1 KiB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 55 bytes 8802 (8.5 KiB)
```

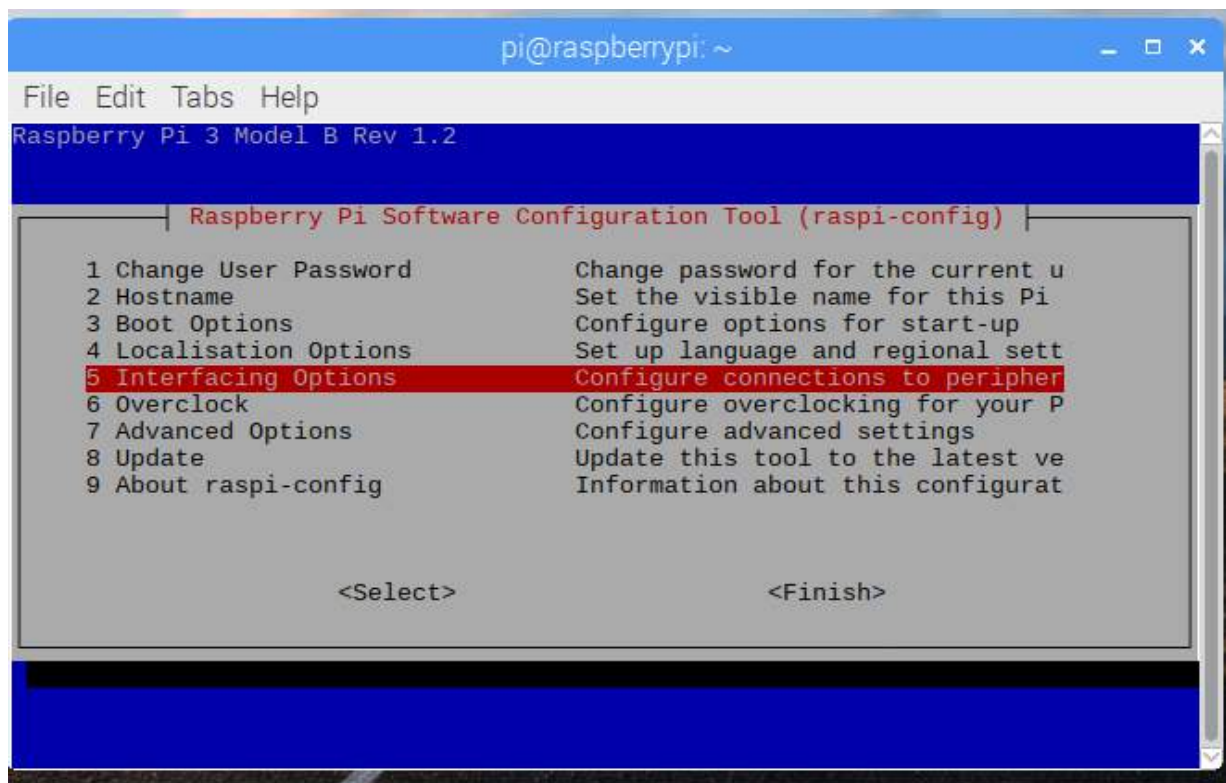
Step 2: Make the setting of SSH connection in raspberry

Type – sudo raspi -config

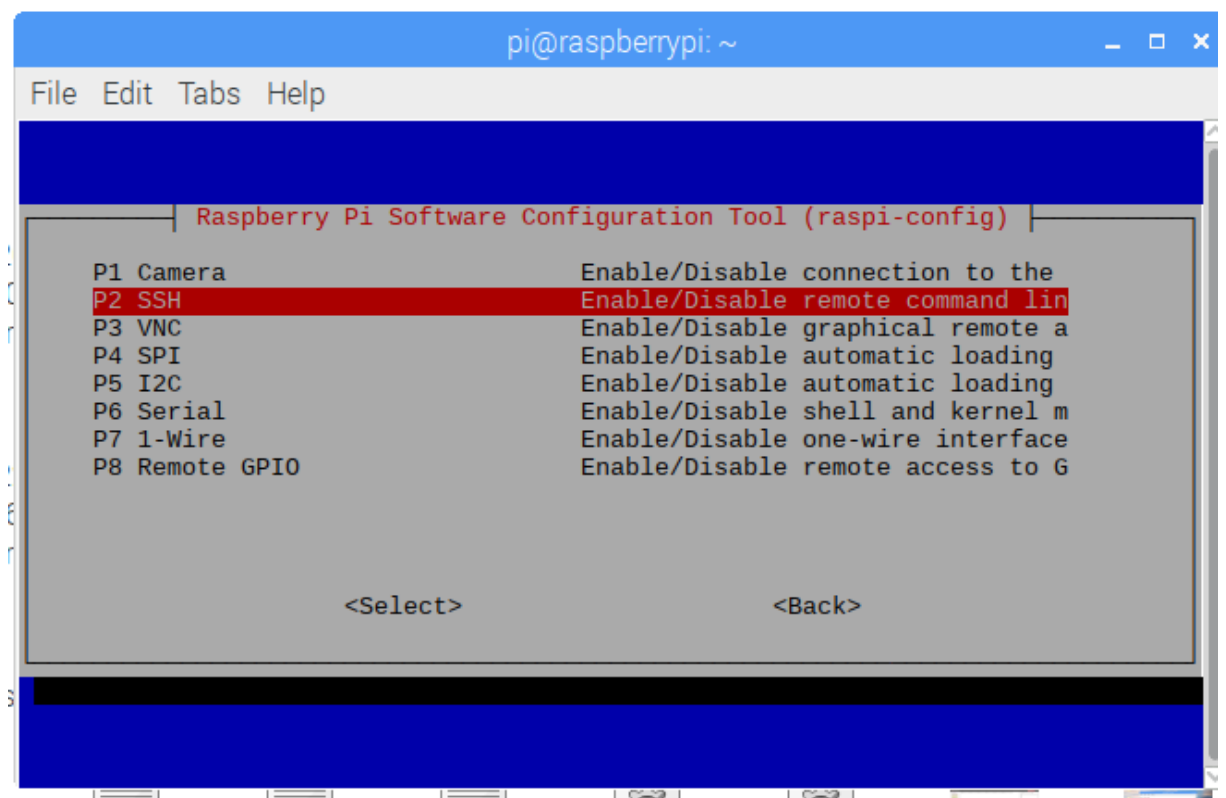


```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~ $ sudo raspi-config
```


Select Interfacing Options



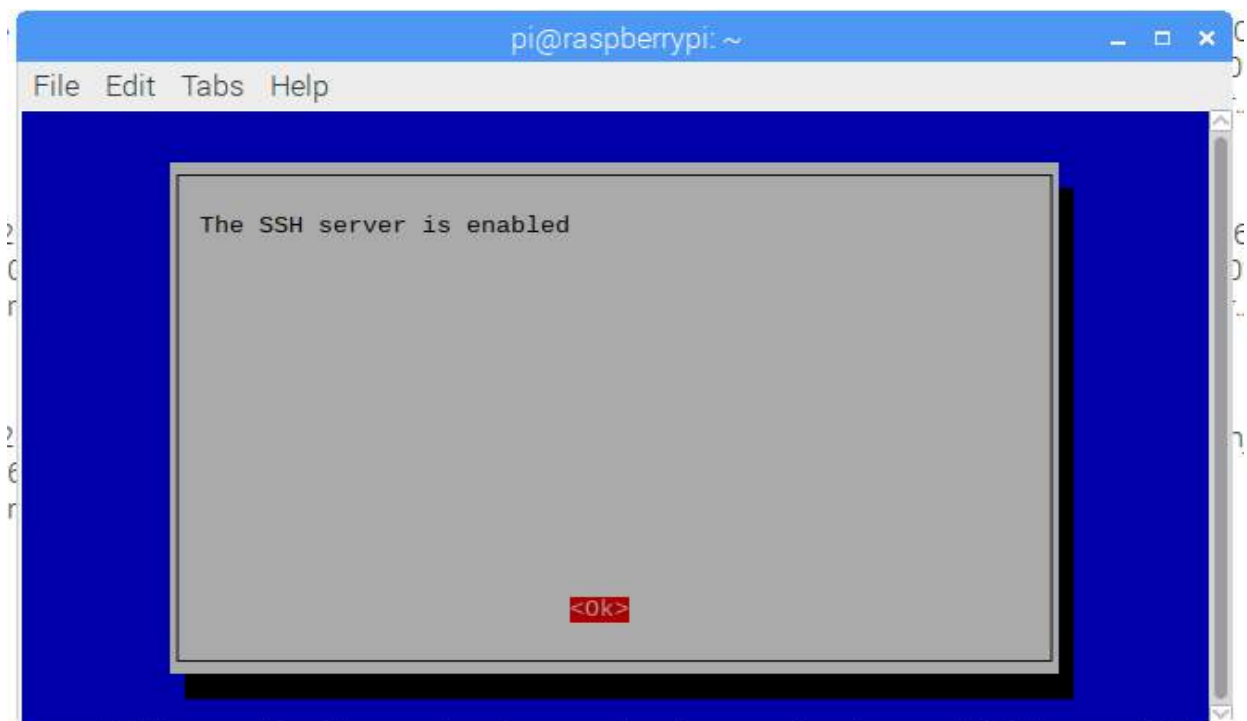
Select SSH



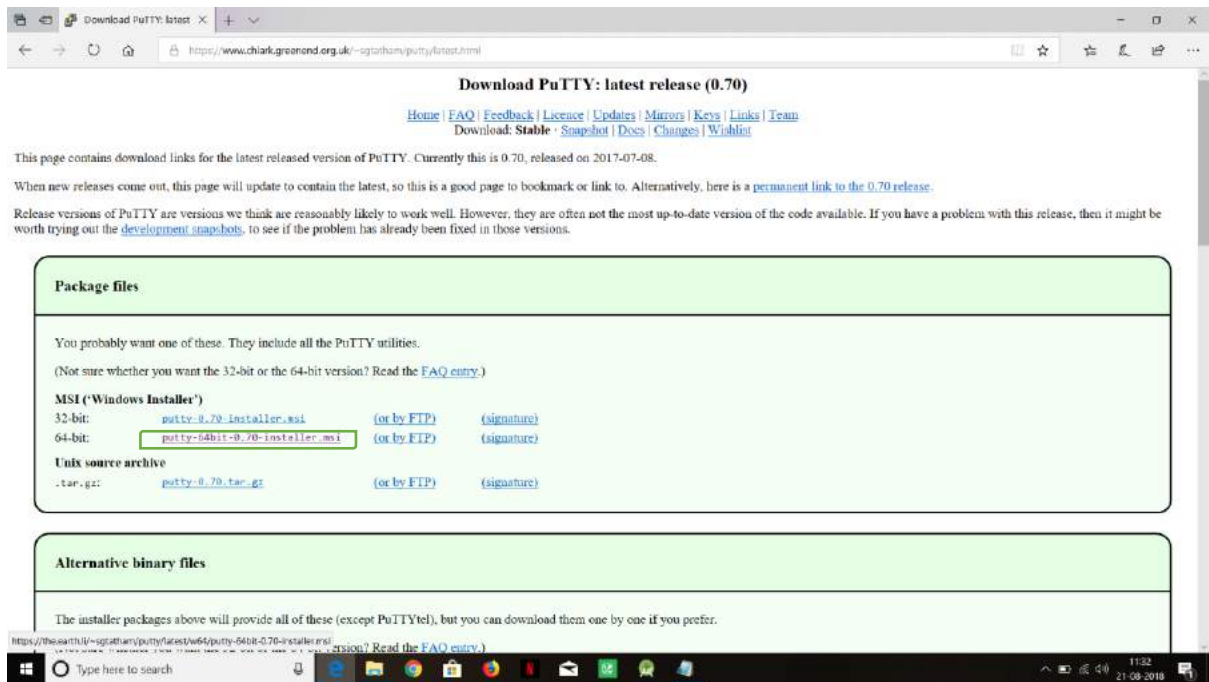
Enable SSH



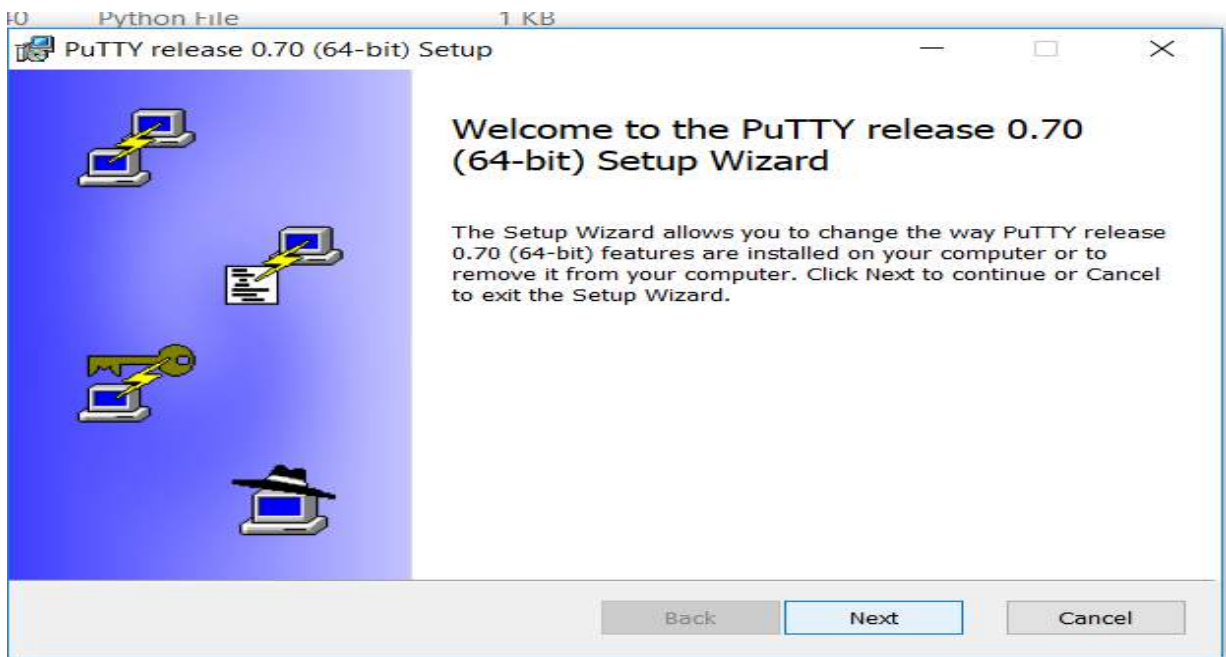
Then Click on OK



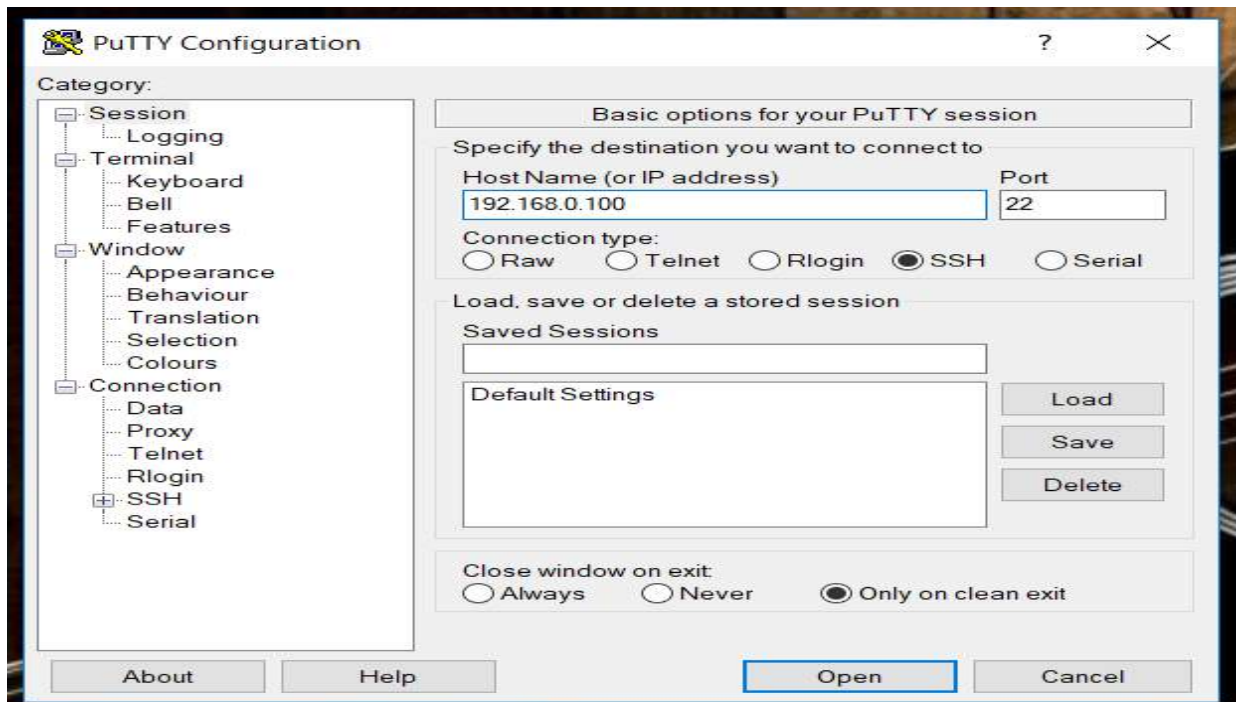
Step 3: Download putty server on your desktop



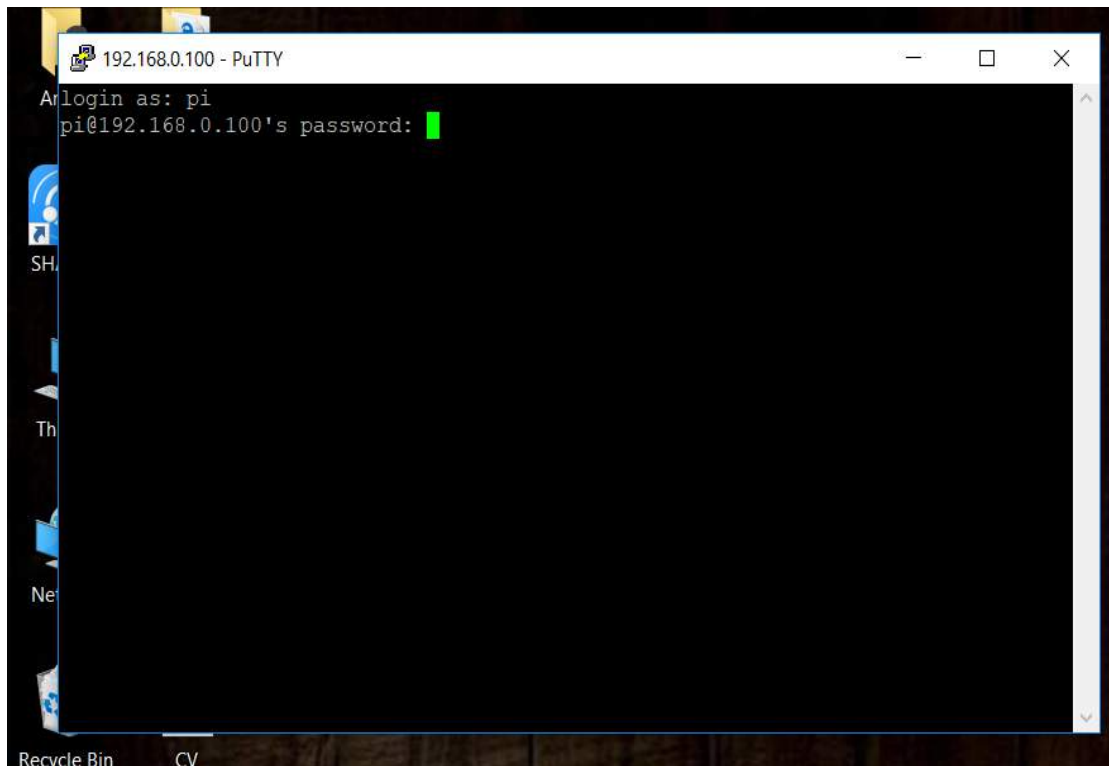
Step 4: Install putty server



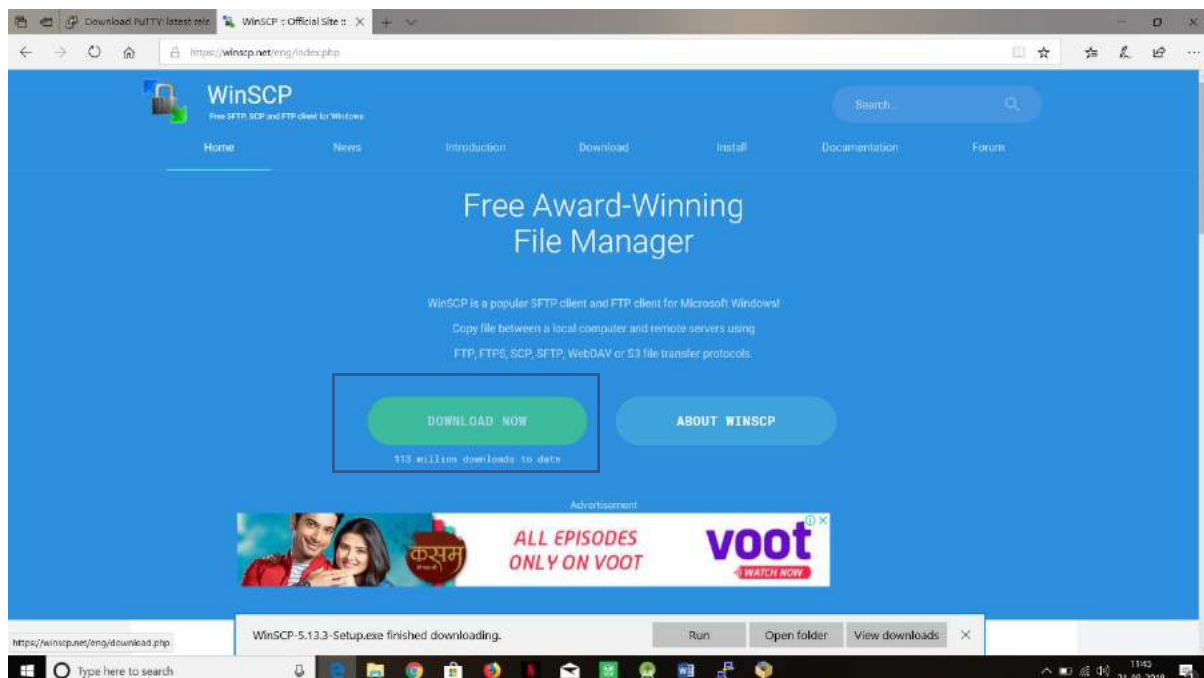
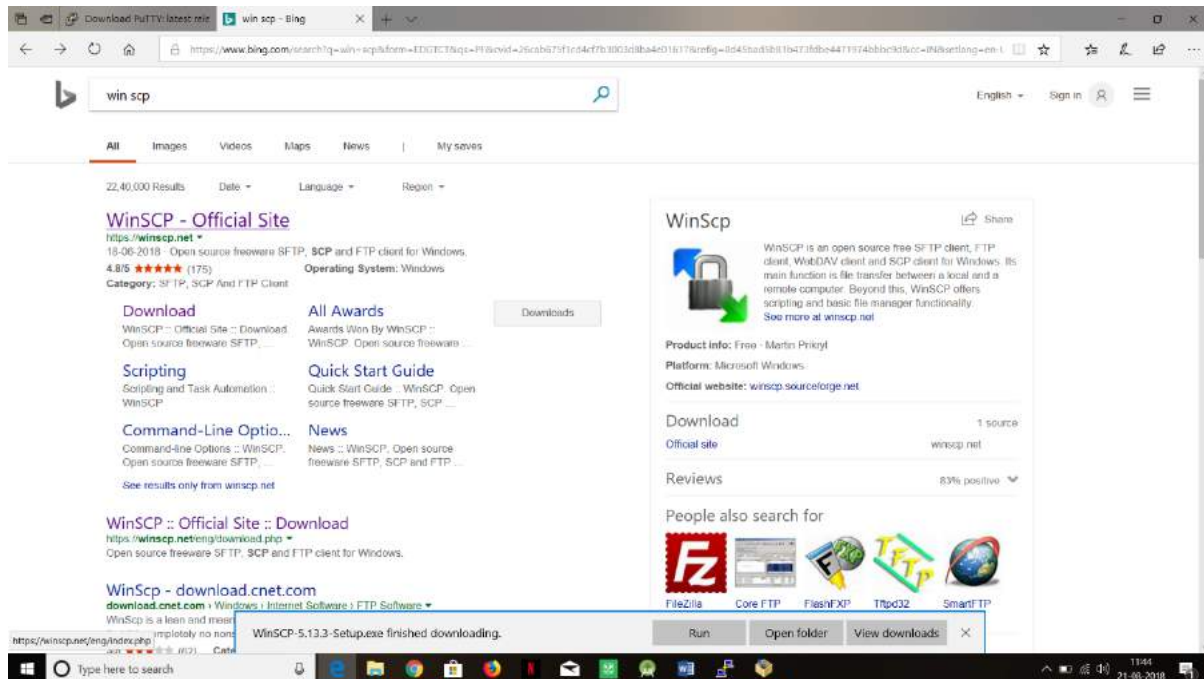
Step 5: Open putty server, then write the Hostname or IP address of raspberry



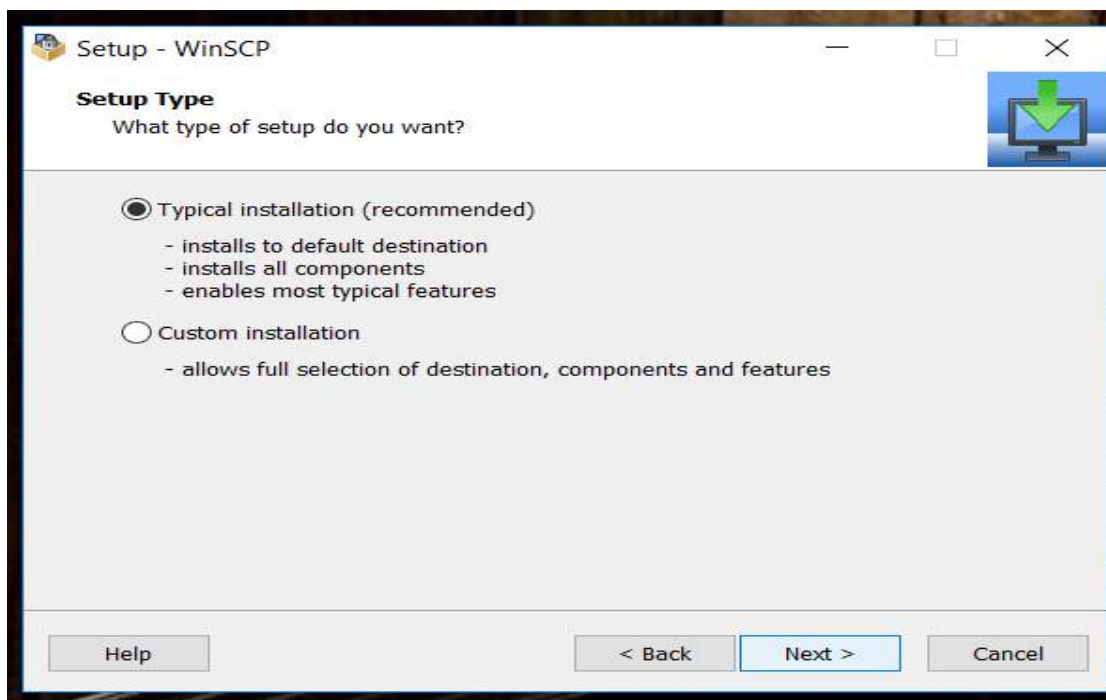
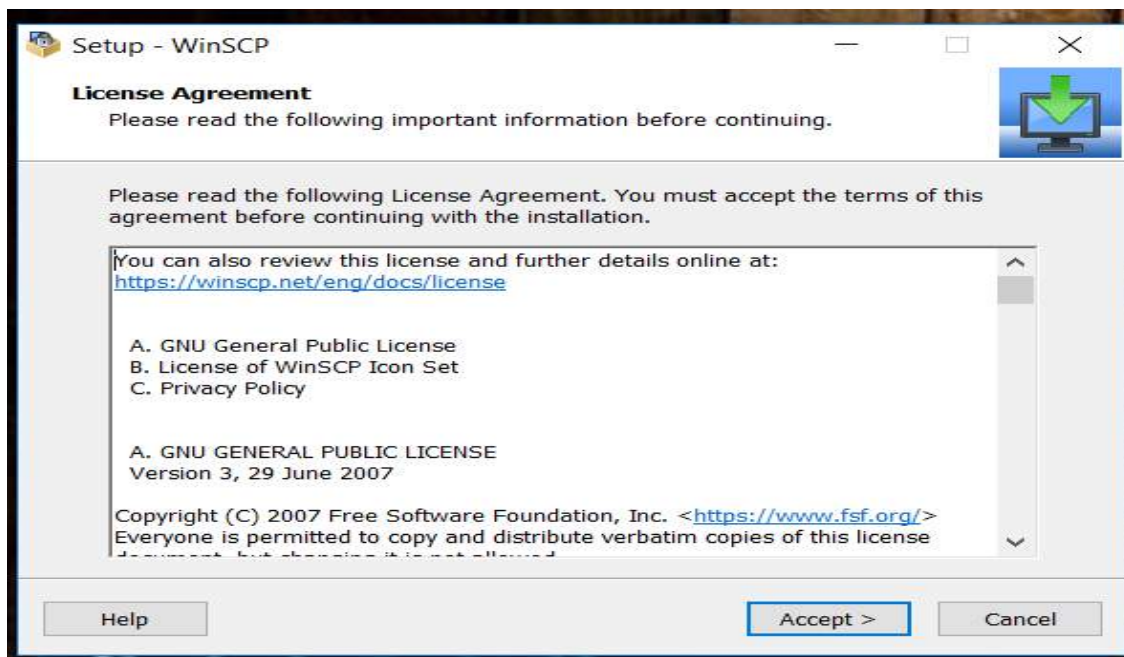
Step 6: Once it is connected it will ask for login & password of raspberry which is “pi” & “raspberry” respectively

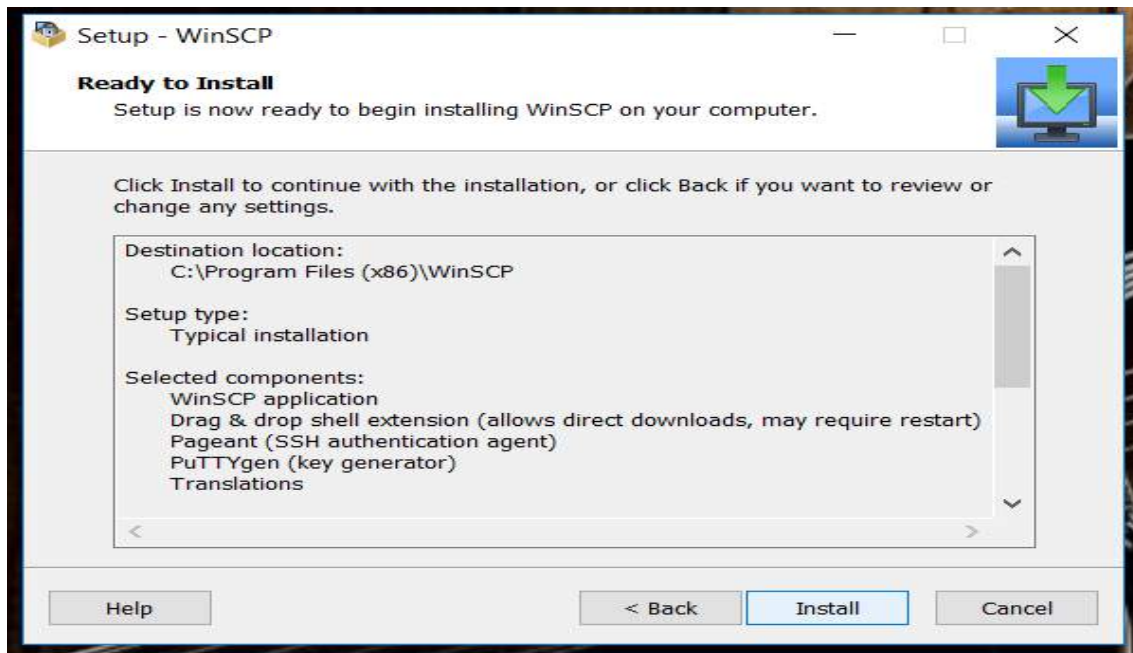


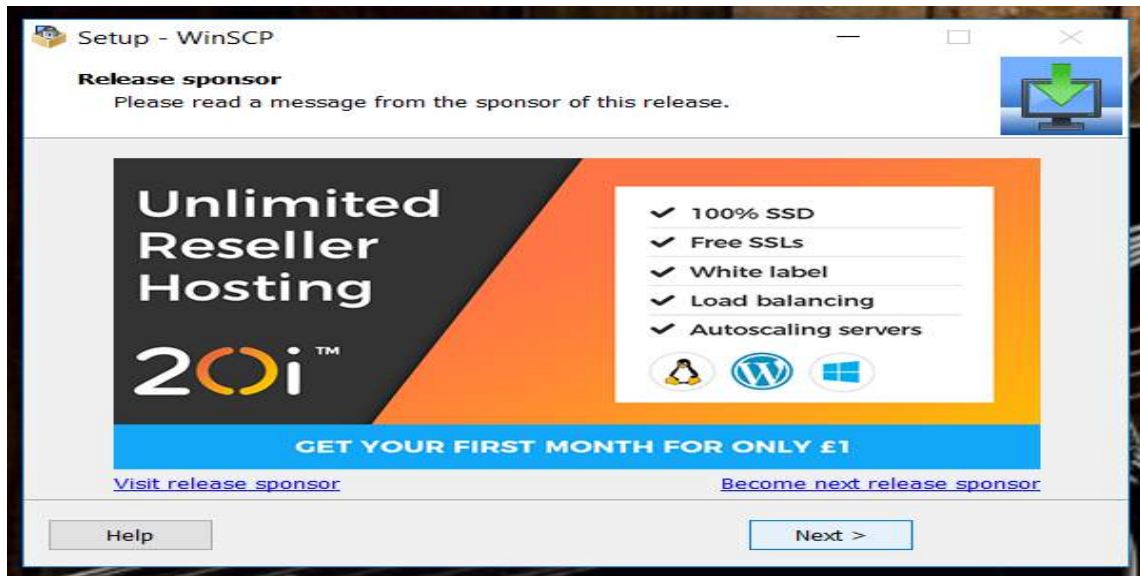
Step 7: Download WinSCP to transfer the file from your desktop to raspberry & vice versa



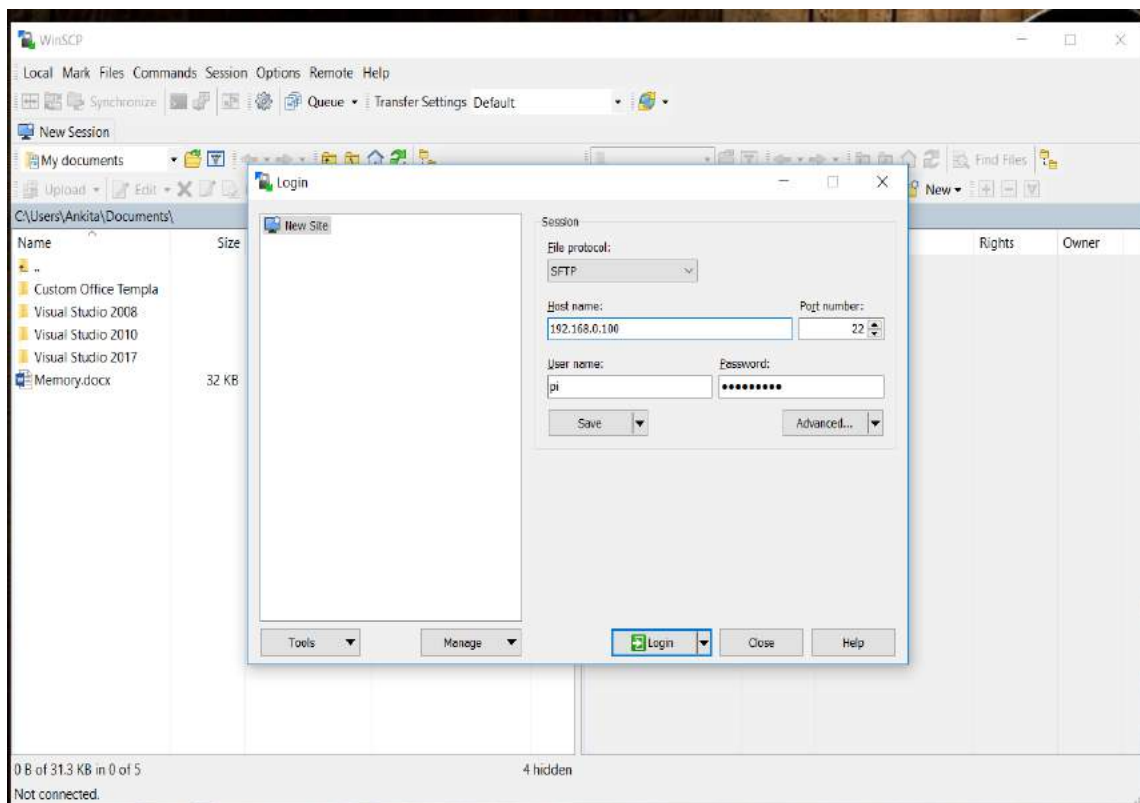
Step 8: Install WinSCP







Step 9: Open WinSCP then enter the IP address, username & password



The screenshot displays the Microsoft Word interface at the top, with the 'Font' ribbon active. Below it, two application windows are open. The first window, titled 'Documents - pi@192.168.0.100 - WinSCP', shows a file explorer view of the local machine's drives. The second window, titled '/home/pi/', shows a terminal output of the 'ls -la' command, displaying a detailed directory listing of the user's home directory, including files like .bashrc, .viminfo, and various project folders.

The screenshot shows the WinSCP interface with the following details:

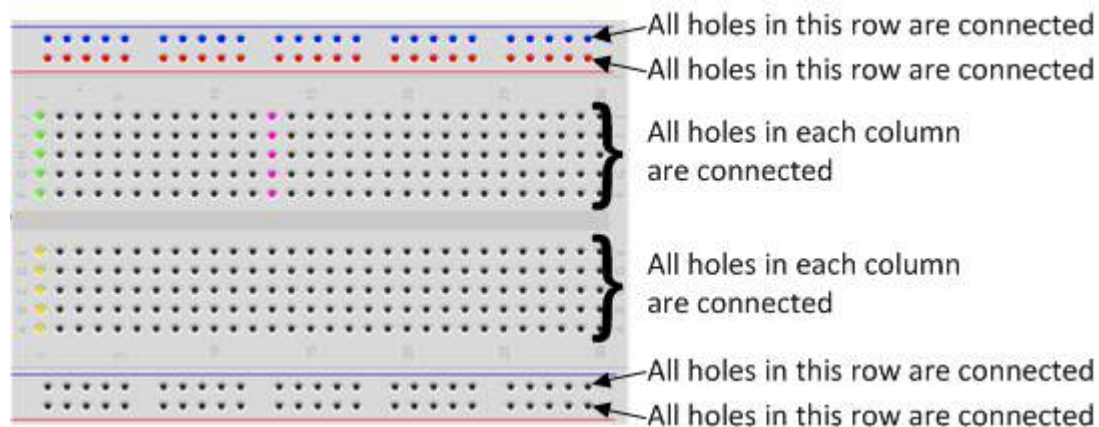
- Title Bar:** pi - pi@192.168.0.100 - WinSCP
- Menu Bar:** Local, Mark, Files, Commands, Session, Options, Remote, Help
- Toolbar:** Synchronize, Queue, Transfer Settings, Default
- Session Bar:** pi@192.168.0.100, New Session
- Local Panel (Left):**
 - Path: C:\Users\Ankita\Documents\
 - Files: .., Custom Office Template, Visual Studio 2008, Visual Studio 2010, Visual Studio 2017, **Memory.docx**, **Photo Album.pptx**
- Remote Panel (Right):**
 - Path: /home/pi/
 - Files: Nodered1.png, Nodered2.png, Nodered3.png, Nodered4.png, Nodered5.png, Nodered7.png, Nodered8.png, Nodered9.png, Nodered10.png, Nodered11.png, Nodered6.png, number, number7, number7, number7.save, panya.py, panya.py.save, **panya.pyz**, **Photo Album.pptx**, pict.jpg, pravya.py, prog.py
- Status Bar:** 0 B of 2.69 MB in 0 of 6, 4 hidden, 0 B of 10.2 MB in 0 of 107, 23 hidden
- Bottom Right:** SFTP-3, 00:25:59

Practical 4:

AIM:GPIO : Light The LED With Python on Breadboard.

The Breadboard

The breadboard is a way of connecting electronic components to each other without having to solder them together. The holes on the breadboard are connected in a pattern.



With the breadboard in the CamJamEduKit, the top row of holes are all connected together – marked with red dots. And so are the second rows of holes – marked with blue dots. The same goes for the two rows of holes at the bottom of the breadboard.

In the middle, the columns of wires are connected together with a break in the middle. So for.

The LED

LED stands for Light Emitting Diode, and glows when electricity is passed through it.

When you pick up the LED, you will notice that one leg is longer than the other. The longer leg (known as the ‘anode’), is always connected to the positive supply of the circuit. The shorter leg (known as the ‘cathode’) is connected to the negative side of the power supply, known as ‘ground’.

The Resistor

You must ALWAYS use resistors to connect LEDs up to the GPIO pins of the Raspberry Pi. The Raspberry Pi can only supply a small current (about 60mA). The LEDs will want to draw more, and if allowed to they will burn out the

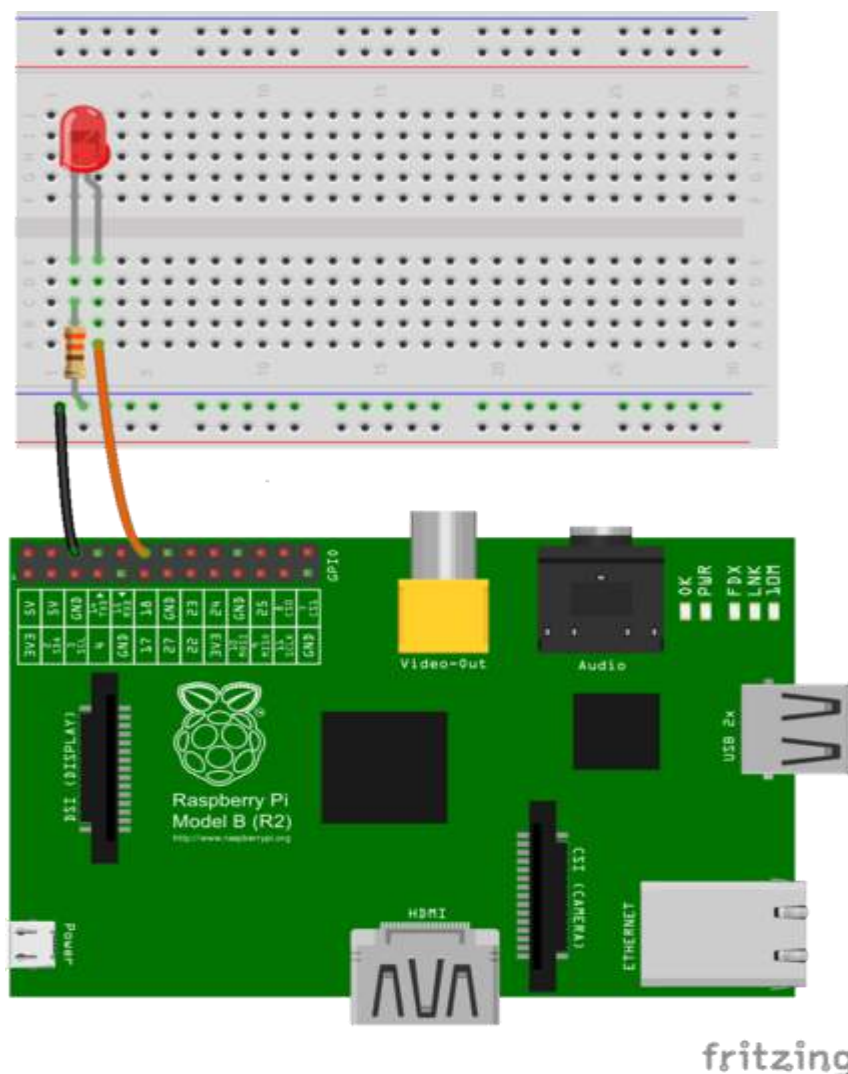
Raspberry Pi. Therefore putting the resistors in the circuit will ensure that only this small current will flow and the Pi will not be damaged.

Jumper Wires

Jumper wires are used on breadboards to ‘jump’ from one connection to another. The ones you will be using in this circuit have different connectors on each end. The end with the ‘pin’ will go into the Breadboard. The end with the piece of plastic with a hole in it will go onto the Raspberry Pi’s GPIO pins.

The Raspberry Pi's GPIO Pins

GPIO stands for **General Purpose Input Output**. It is a way the Raspberry Pi can control and monitor the outside world by being connected to electronic circuits.



You should turn your Pi off for the next bit, just in case you accidentally short something out.

- Use one of the jumper wires to connect a ground pin to the rail, marked with blue, on the breadboard. The female end goes on the Pi's pin, and the male end goes into a hole on the breadboard.
- Then connect the resistor from the same row on the breadboard to a column on the breadboard, as shown above.
- Next, push the LEDs legs into the breadboard, with the long leg (with the kink) on the right.

Write a Python Code to glow an LED on (Using BOARD mode) :->

Step 1 : Source Code

Nano LED.py

```
import time
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BOARD)
GPIO.setup(8, GPIO.OUT)
GPIO.output(8,False)
while True:
    GPIO.output(8,False)
```

Once you have typed all the code and checked it, save and exit the text editor with “Ctrl + x” then “y” then “enter”.

Step 2 : Running the Code

To run this code type:

sudo python LED.py

You will see the LED turn on for a second and then turn off.

Explanation

<code>import RPi.GPIO as GPIO</code>	The first line tells the Python interpreter (the thing that runs the Python code) that it will be using a 'library' that will tell it how to work with the Raspberry Pi's GPIO pins. A 'library' gives a programming language extra commands that can be used to do something different that it previously did not know how to do. This is like adding a new channel to your TV so you can watch something different.
<code>import time</code>	Imports the Time library so that we can pause the script later on.
<code>GPIO.setmode(GPIO.BOARD)</code>	Each pin on the Pi has several different names, so you need to tell the program which naming convention is to be used.
<code>GPIO.setwarnings(False)</code>	This tells Python not to print GPIO warning messages to the screen.
<code>GPIO.setup(8, GPIO.OUT)</code>	This line tells the Python interpreter that pin 18 is going to be used for outputting information, which means you are going to be able to turn the pin 'on' and 'off'.
<code>GPIO.output(8,True)</code>	This turns the GPIO pin 'on'. What this actually means is that the pin is made to provide power of 3.3volts. This is enough to turn the LED in our circuit on.
<code>time.sleep(1)</code>	Pauses the Python program for 1 second
<code>GPIO.output(8,False)</code>	This turns the GPIO pin 'off', meaning that the pin is no longer supplying any power.

The output look like below image :



Write a Python Code to glow an LED on (Using BCM mode) :->

```
import time
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(23, GPIO.OUT)
GPIO.output(23,False)
while True:
    GPIO.output(23,False)
```

Write a program to blink LED using BOARD mode . Take pin as a input from user :->

```
import time
import RPi.GPIO as GPIO
a=int(input("Enter pin number:"))
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(a, GPIO.OUT)
GPIO.output(a,True)
```

```
while True:
    GPIO.output(a, False)
    time.sleep(1)
    GPIO.output(a, True)
    time.sleep(1)
```

Write a program to blink LED. Take time as a input from user (using BOARD mode):->

```
import time
import RPi.GPIO as GPIO
a=float(input("Enter time:"))
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(11, GPIO.OUT)
GPIO.output(11,True)
```

```
while True:
    GPIO.output(11, False)
    time.sleep(a)
    GPIO.output(11, True)
    time.sleep(a)
```

Write a program to blink LED. Take pin, time as a input from user. (Using BOARD mode):->

```
import time
import RPi.GPIO as GPIO
a=int(input("Enter pin number:"))
b=float(input("Enter time:"))
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(a, GPIO.OUT)
GPIO.output(a,True)
```

```
while True:
    GPIO.output(a,False)
    time.sleep(b)
    GPIO.output(a, True)
    time.sleep(b)
```

Write a program to blink the LED number of times requested by user (Using BCM Mode) :->

```
import time
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(23, GPIO.OUT)
GPIO.output(23, True)
y = int(input("Enter Number of times to blink an LED :"))
for i in range(0,y):
    GPIO.output(23, False)
    time.sleep(1)
    GPIO.output(23, True)
    time.sleep(1)
```

Write a program to Glow the LED .If condition is satisfied. Take choice as a input from user. (Using if-elif loop) :->

```
import time
import RPi.GPIO as GPIO
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(8, GPIO.OUT)
GPIO.output(8, True)
y = int(input("Enter your choice 1.To Glow an LED 2. To switch off an LED
3. To blink an LED:"))
if (y==1):
    a=int(input("Enter a pin number:"))
    while True:
        GPIO.output(8, True)
elif (y==2):
    b=int(input("Enter a pin number:"))
    while False:
        GPIO.output(8, False)
elif (y==3):
    while True:
        c=float(input("Enter time:"))
        GPIO.output(8, True)
        time.sleep(c)
        GPIO.output(8, True)
        time.sleep(c)
```

Write a program to blink 3 LED's 8 times :->

```
import time
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BOARD)
GPIO.setup(7, GPIO.OUT)
GPIO.output(7,True)
GPIO.setup(8, GPIO.OUT)
GPIO.output(8,True)
GPIO.setup(10, GPIO.OUT)
GPIO.output(10,True)
for i in range(1,9);
    GPIO.output(7,False)
    time.sleep(1)
    GPIO.output(7,True)
    time.sleep(1)
    GPIO.output(8,False)
    time.sleep(1)
    GPIO.output(8,True)
    time.sleep(1)
    GPIO.output(10,False)
    time.sleep(1)
    GPIO.output(10,True)
    time.sleep(1)
```

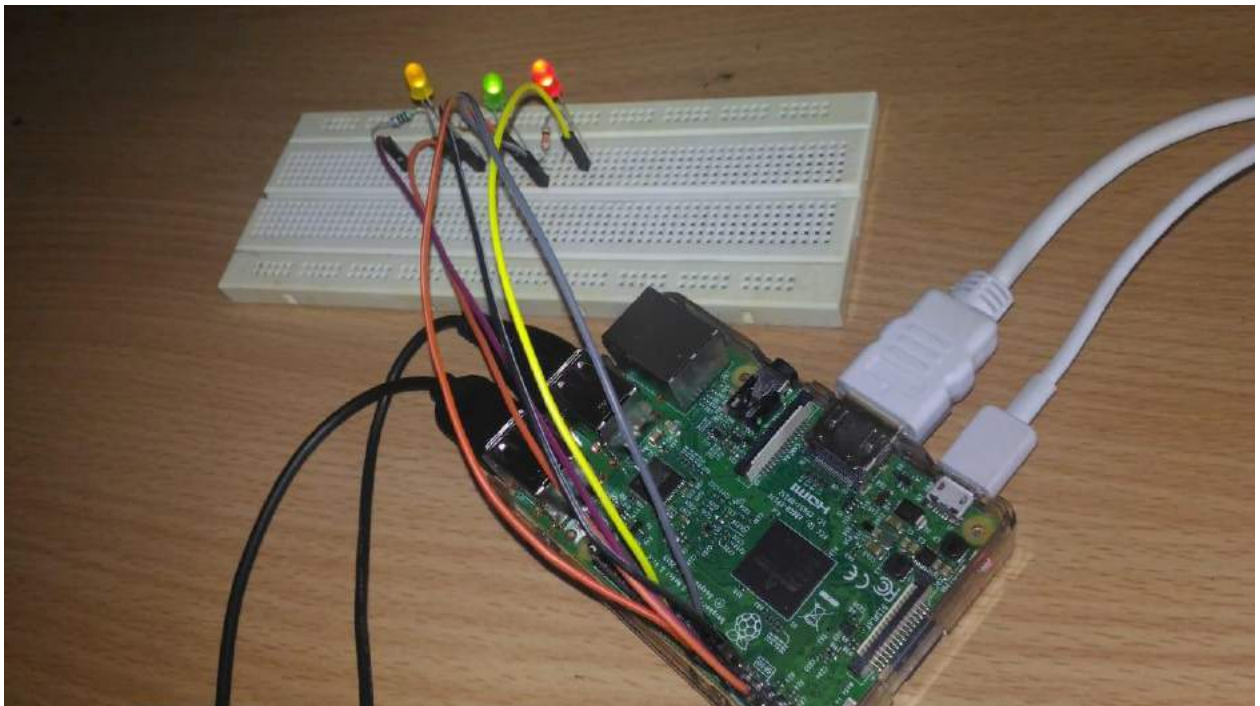
Write a program to blink 3 LED's. Take time as a input from user :->

```
import time
import RPi.GPIO as GPIO
a=float(input("Enter time:"))
b=float(input("Enter time:"))
c=float(input("Enter time:"))
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(8, GPIO.OUT)
GPIO.setup(11, GPIO.OUT)
GPIO.setup(15, GPIO.OUT)
while True:
    GPIO.output(8, False)
    time.sleep(a)
    GPIO.output(8, True)
```



```
time.sleep(a)
GPIO.output(11, False)
time.sleep(b)
GPIO.output(11, True)
time.sleep(b)
GPIO.output(15, False)
time.sleep(c)
GPIO.output(15, True)
time.sleep(c)
```

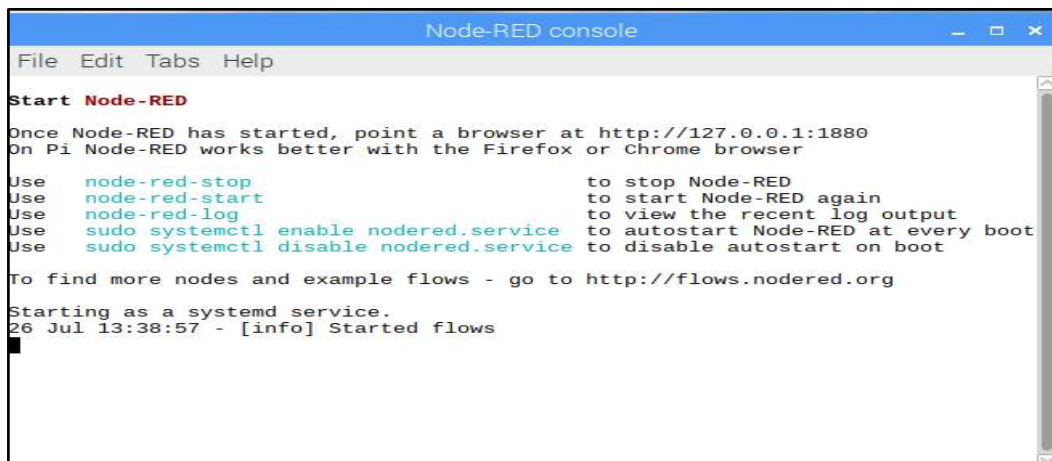
The output look like below image :



Practical 5:

AIM: Run the node red editor and run simple programs and trigger GPIO.
Use basic nodes such as inject, debug, GPIO.

1.Copy the below given link



```
Node-RED console
File Edit Tabs Help

Start Node-RED

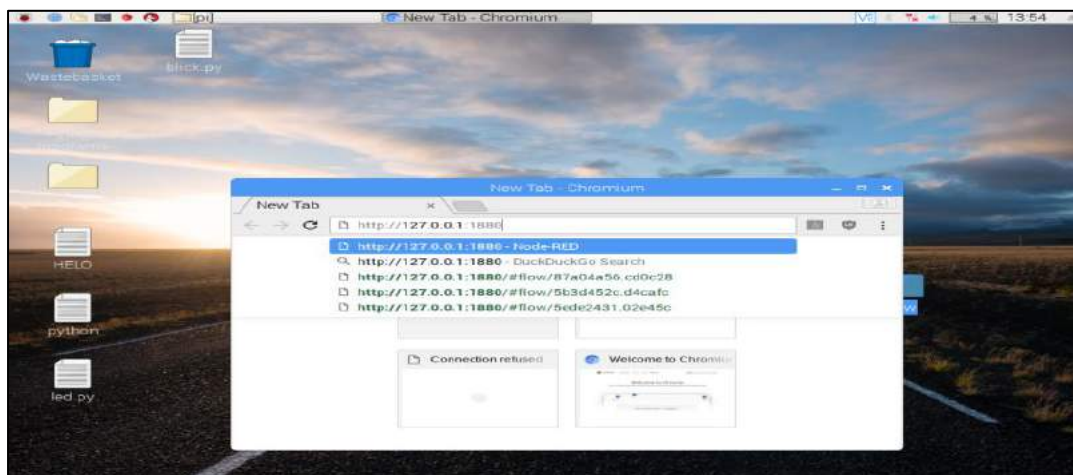
Once Node-RED has started, point a browser at http://127.0.0.1:1880
On Pi Node-RED works better with the Firefox or Chrome browser

Use node-red-stop to stop Node-RED
Use node-red-start to start Node-RED again
Use node-red-log to view the recent log output
Use sudo systemctl enable nodered.service to autostart Node-RED at every boot
Use sudo systemctl disable nodered.service to disable autostart on boot

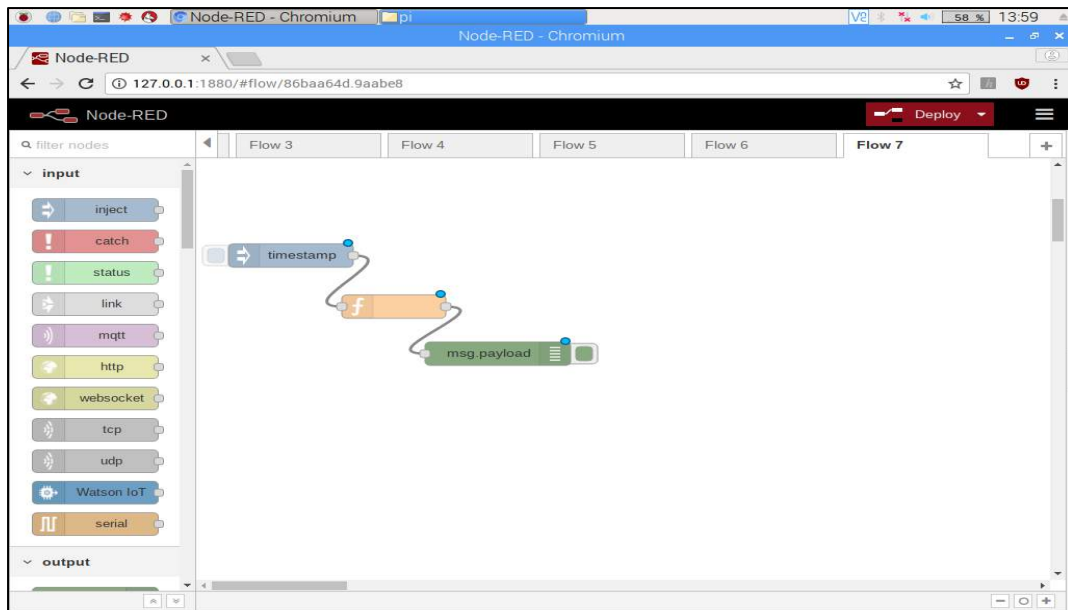
To find more nodes and example flows - go to http://flows.nodered.org

Starting as a systemd service.
26 Jul 13:38:57 - [info] Started flows
```

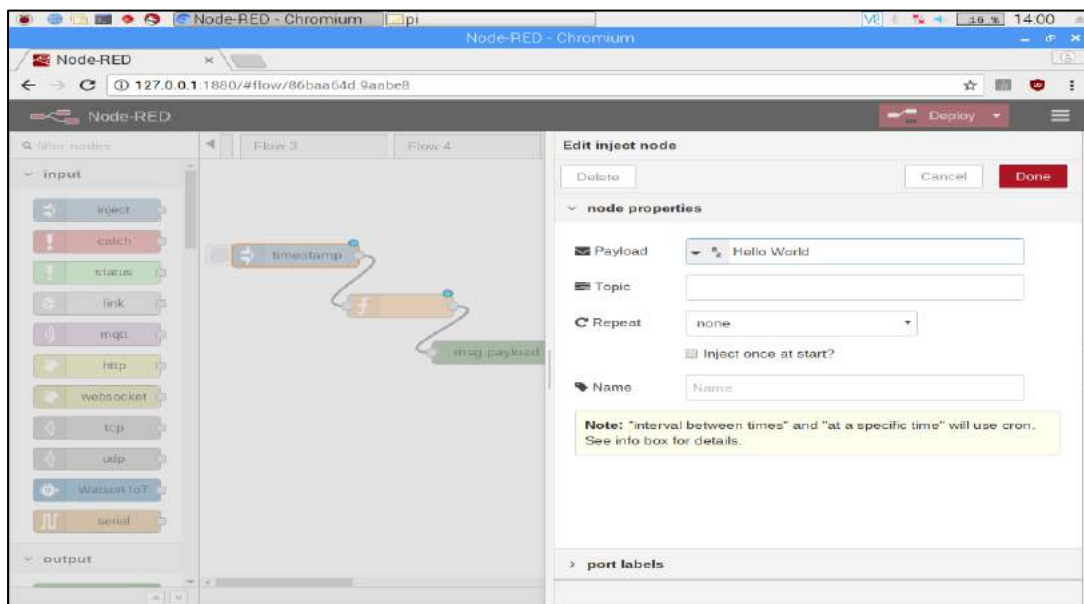
2. Paste it in the browser



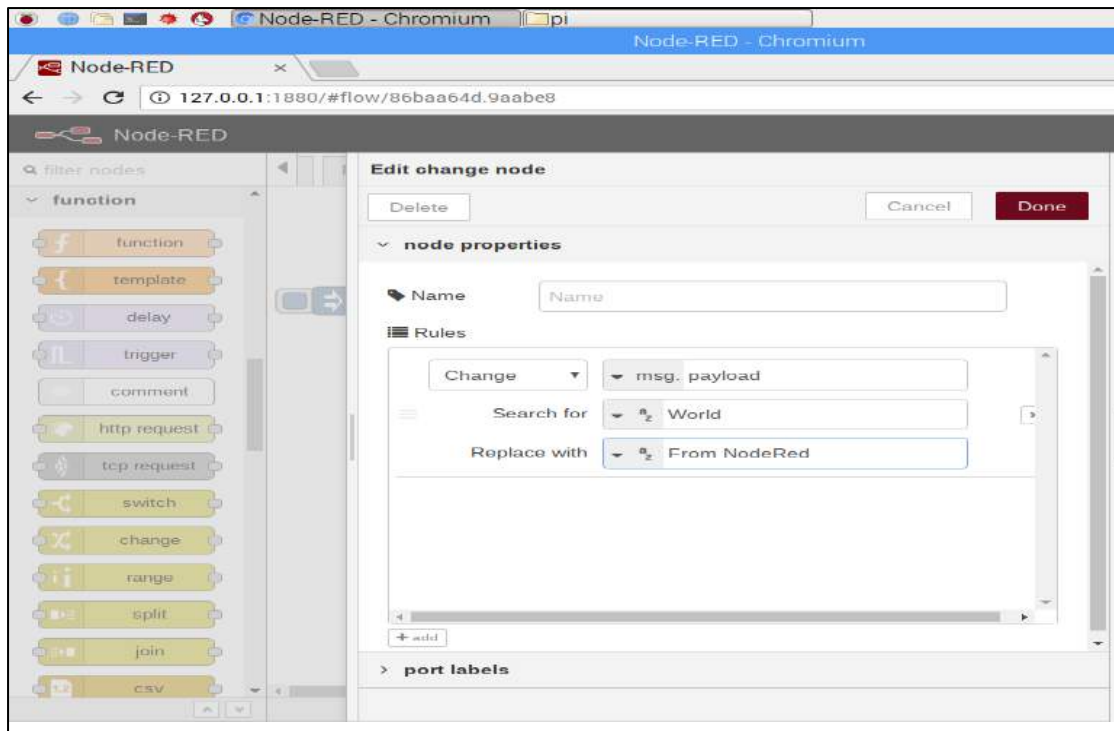
3. There are some readymade functions given in node red Eg:Change function. Create a flow as given below



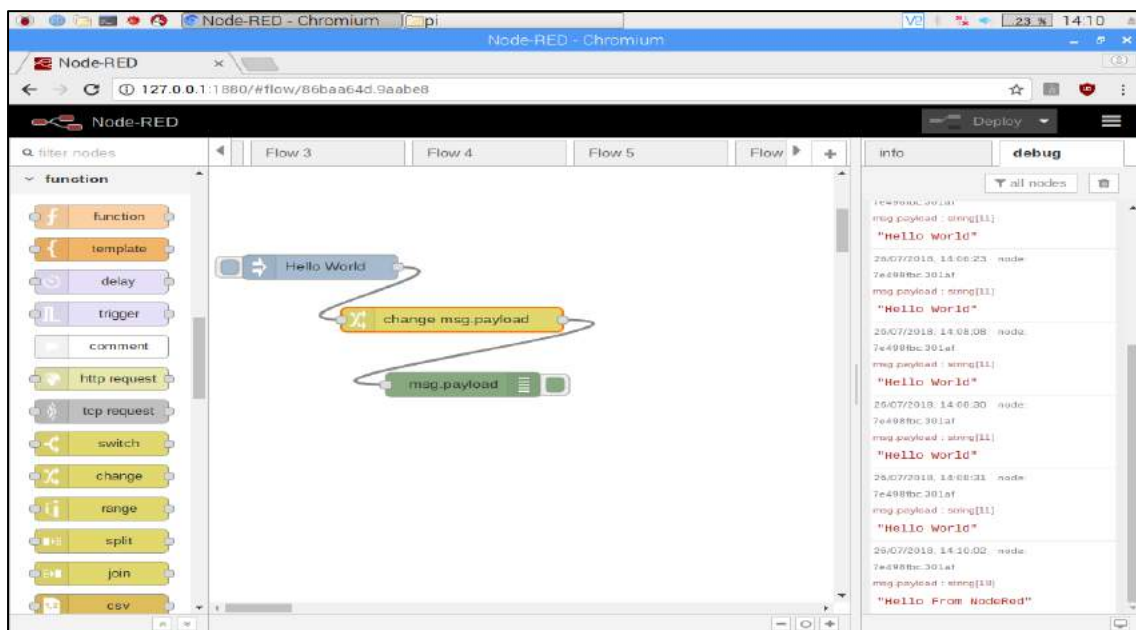
4. Double click on time stamp and write it as “Hello World” in payload



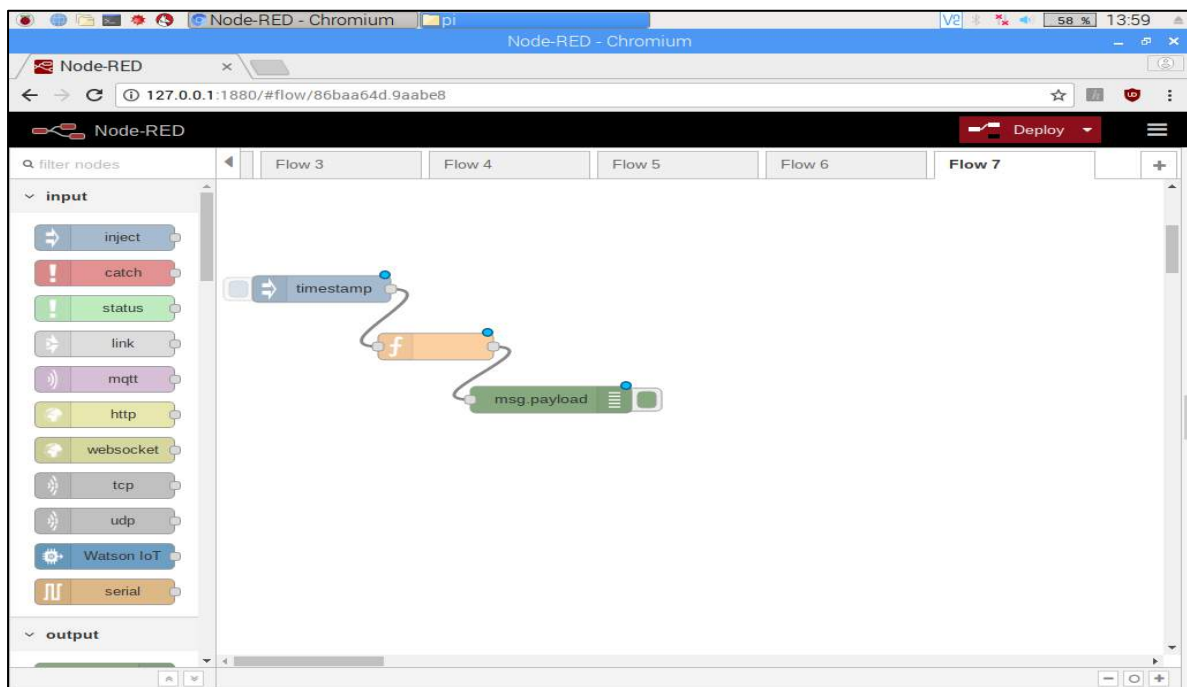
5. Double click on change function and search for “world” and replace with “From Nodered”.



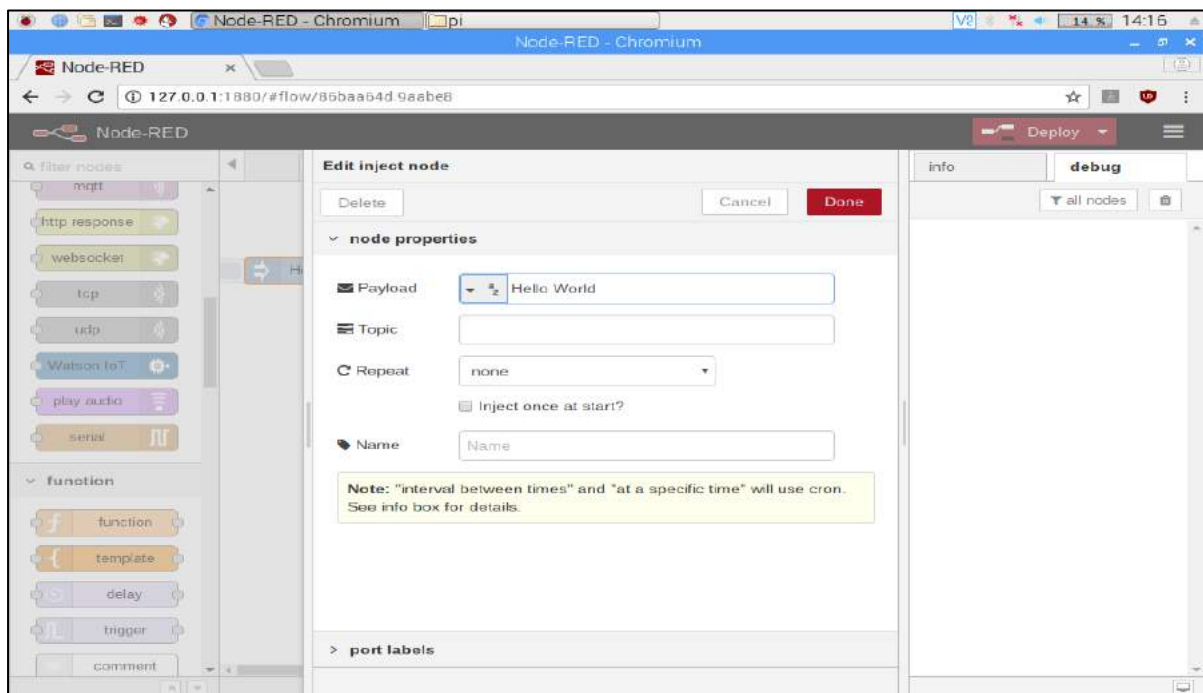
6. Then deploy and check the result.



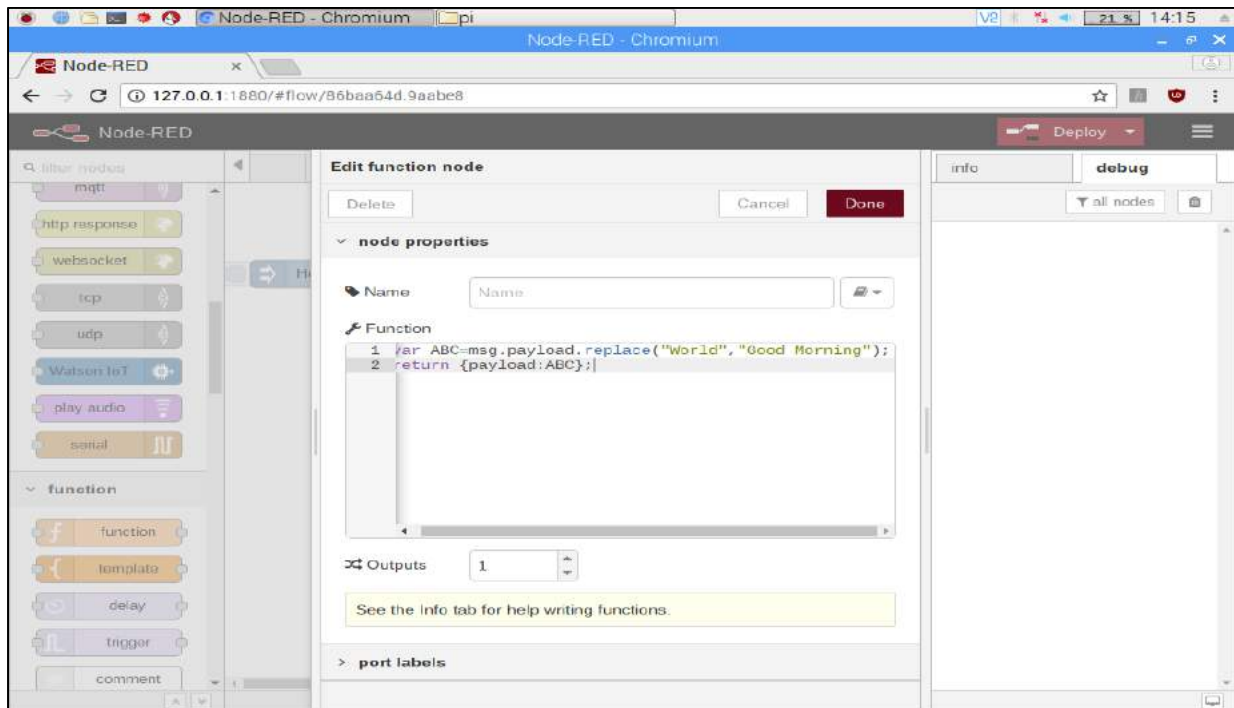
6. By using self declared functions(change function): Create a similar flow as above but instead of using readymade change function use a simple function



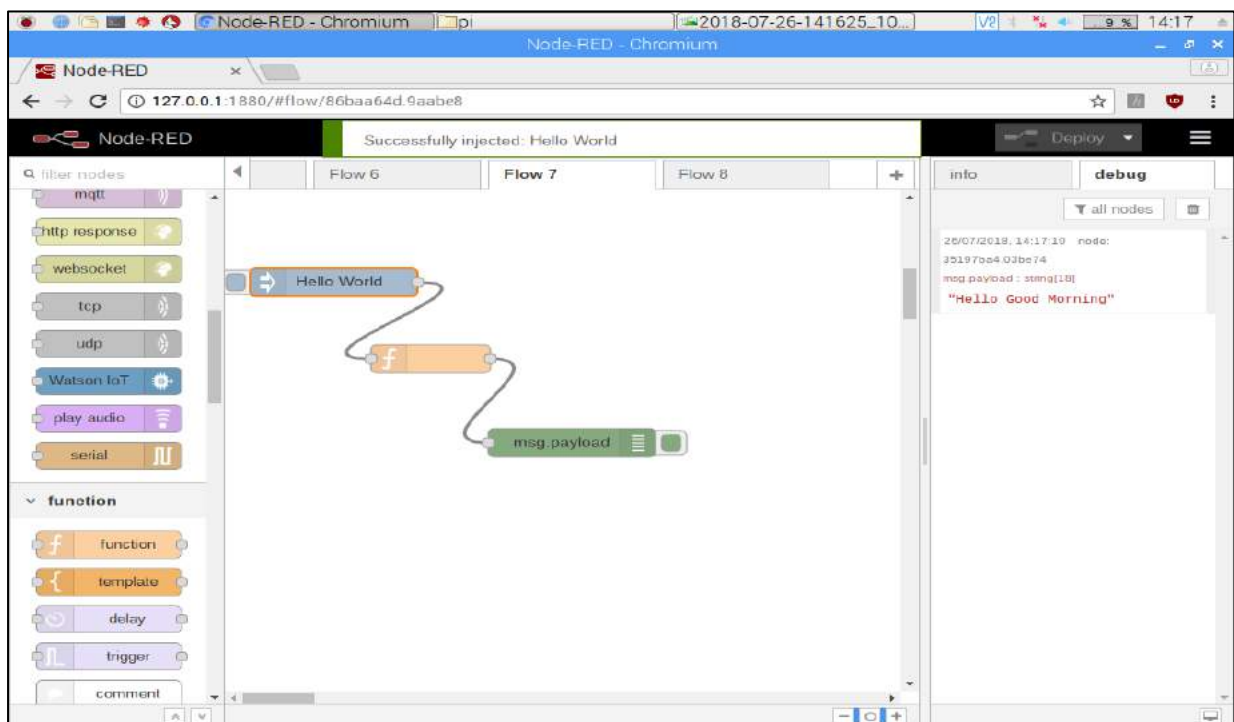
8. Double click on time stamp and write it as “Hello World” in payload



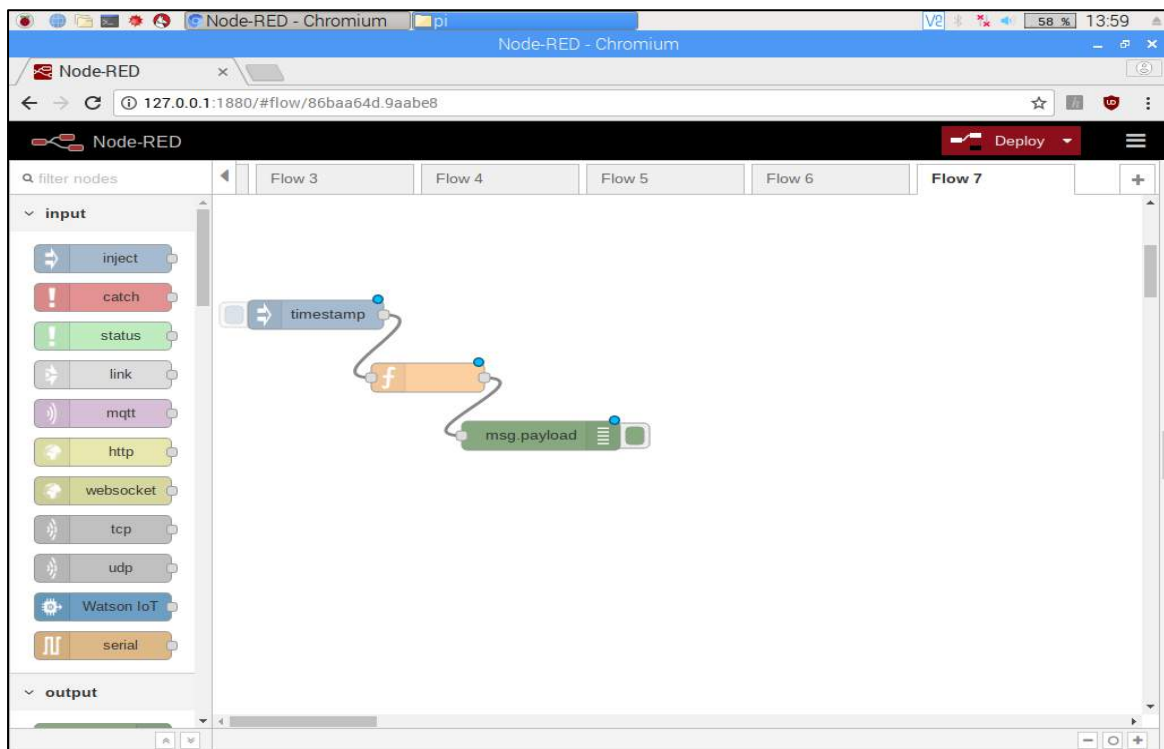
9. Double click on function and write the below code



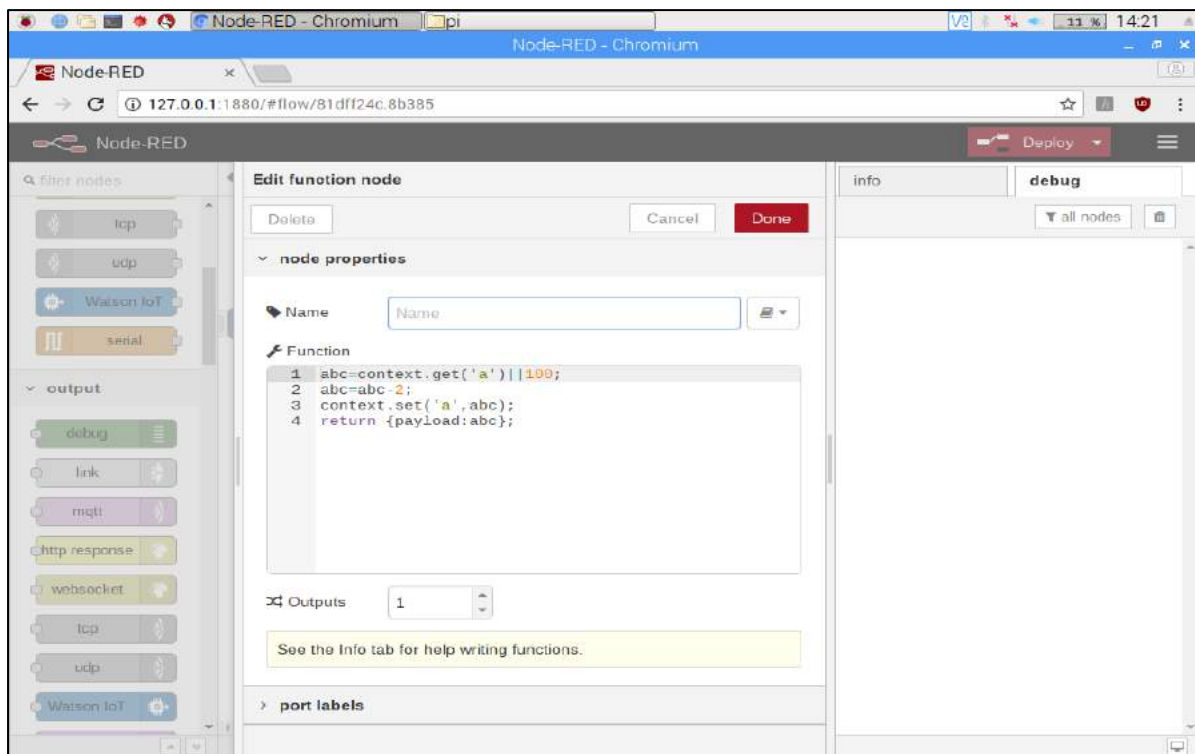
10. Deploy and check the output



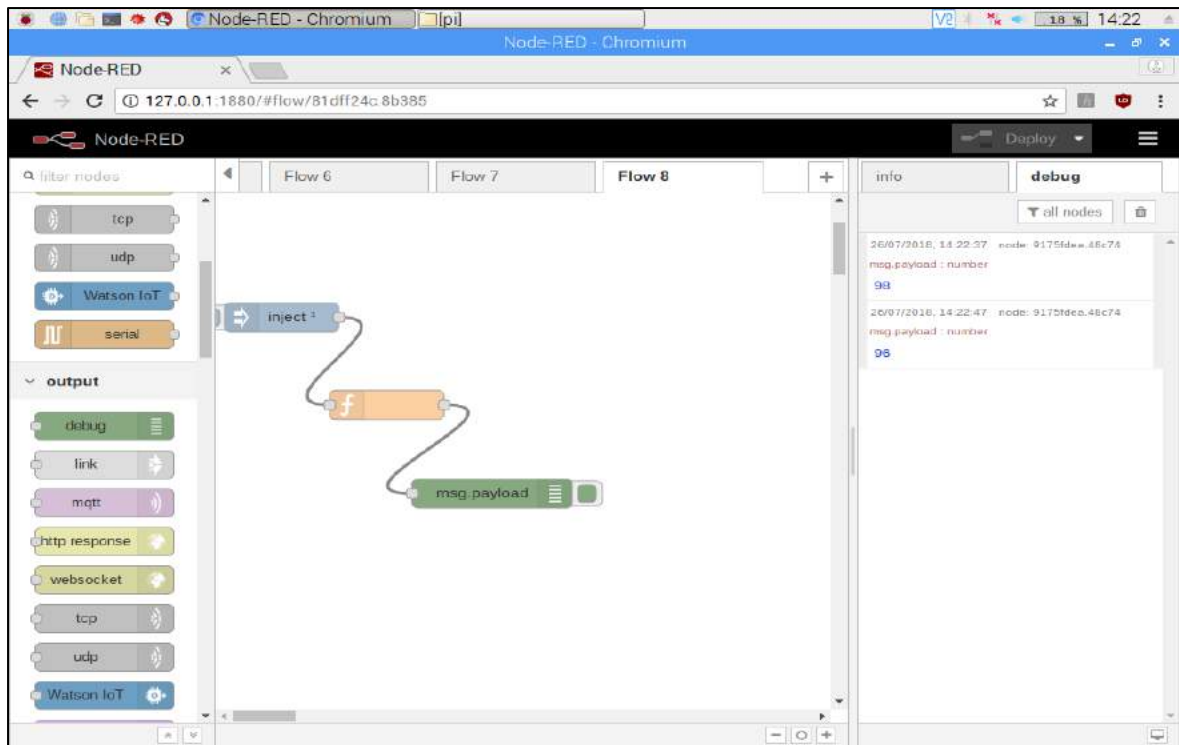
11. Function to reduce the number by 2. Create the flow as below



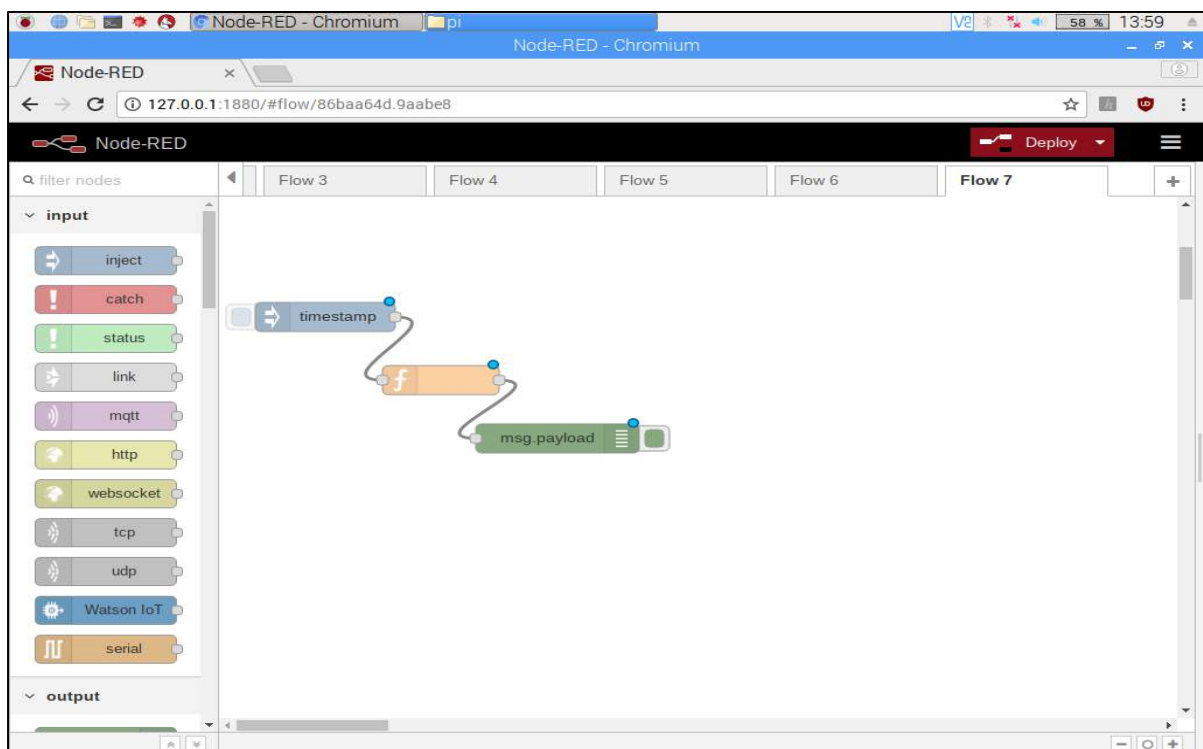
12. Double click the function node and write the code given below:



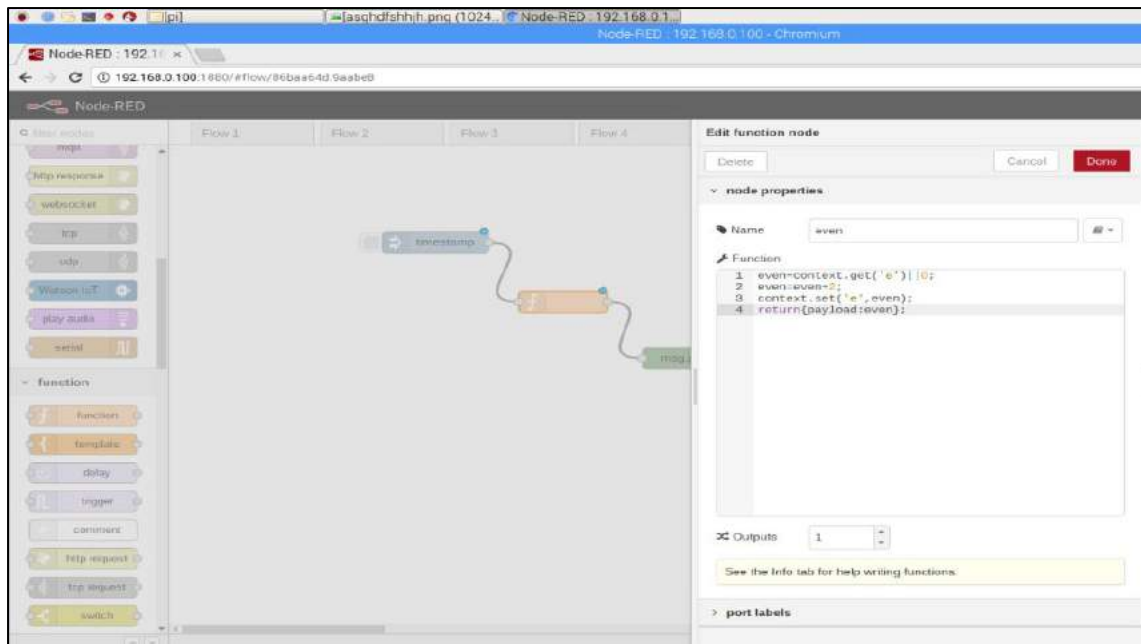
13. Then deploy and check the output



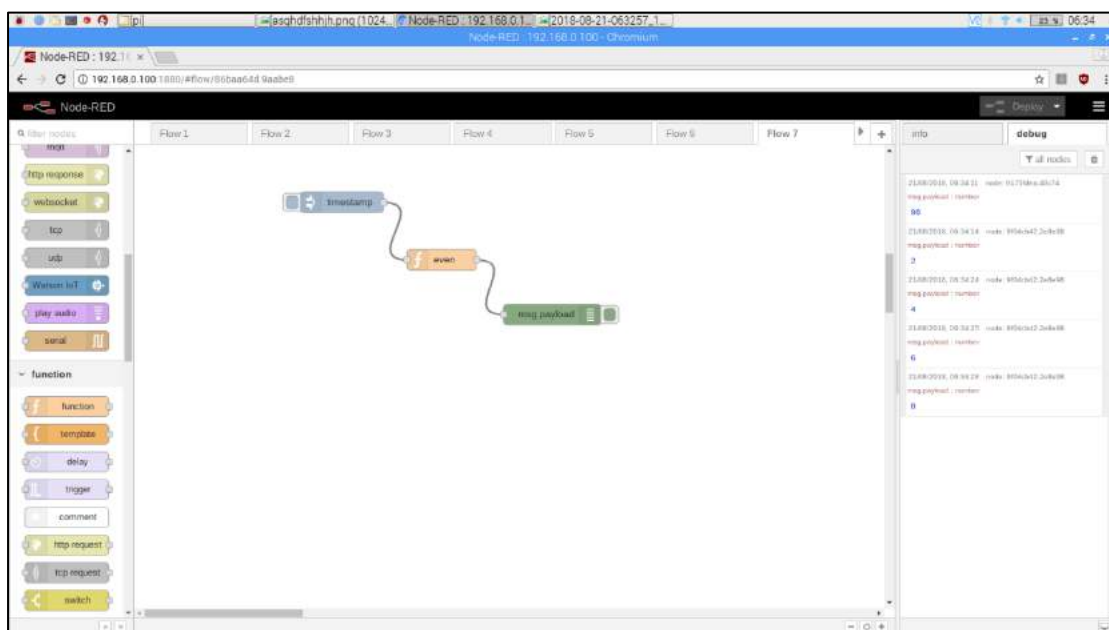
14. Function to display even numbers. Create the flow as below



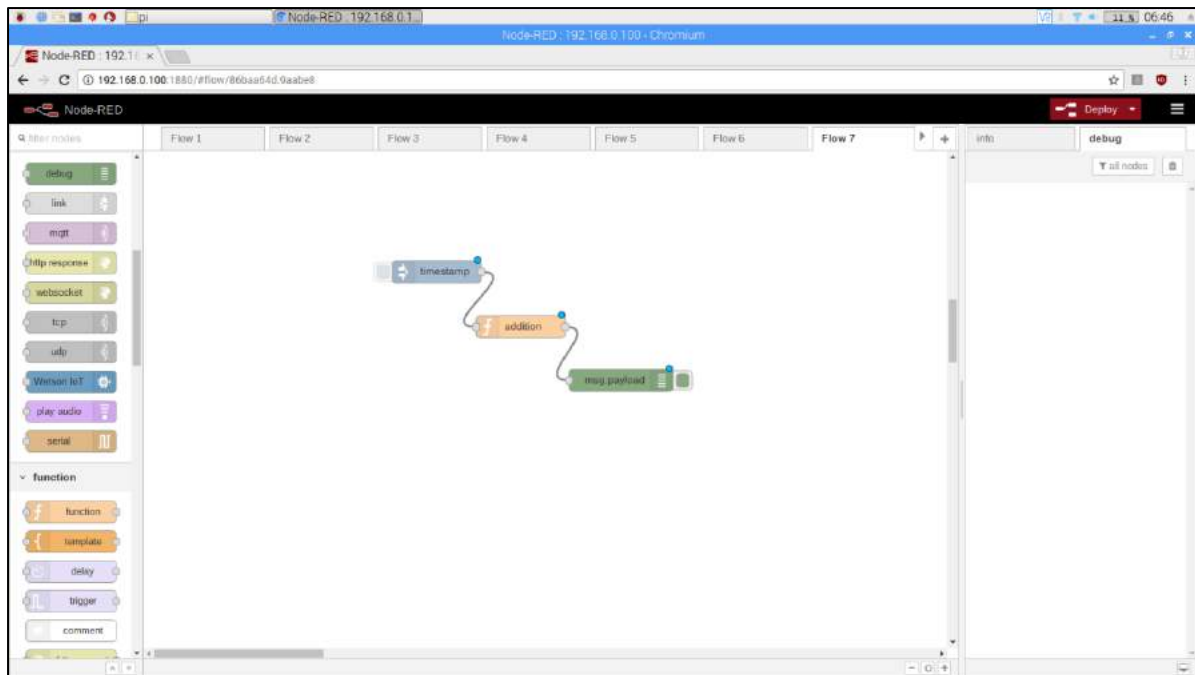
15. Double click the function and write the code given below:



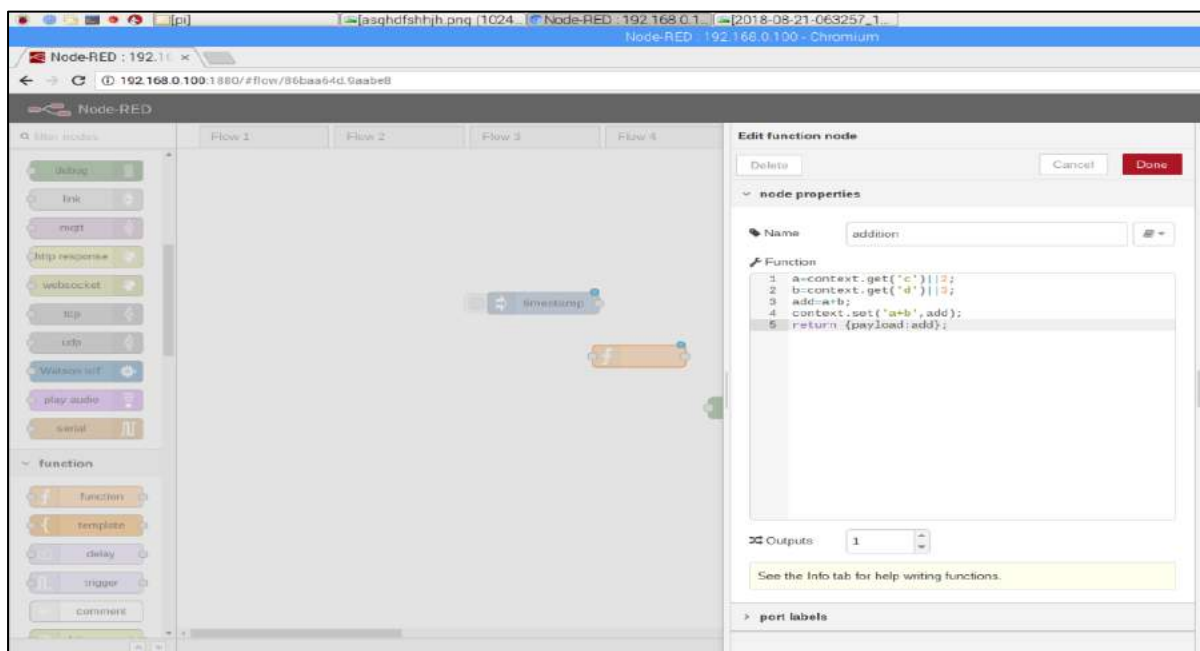
16. Deploy and check the output.



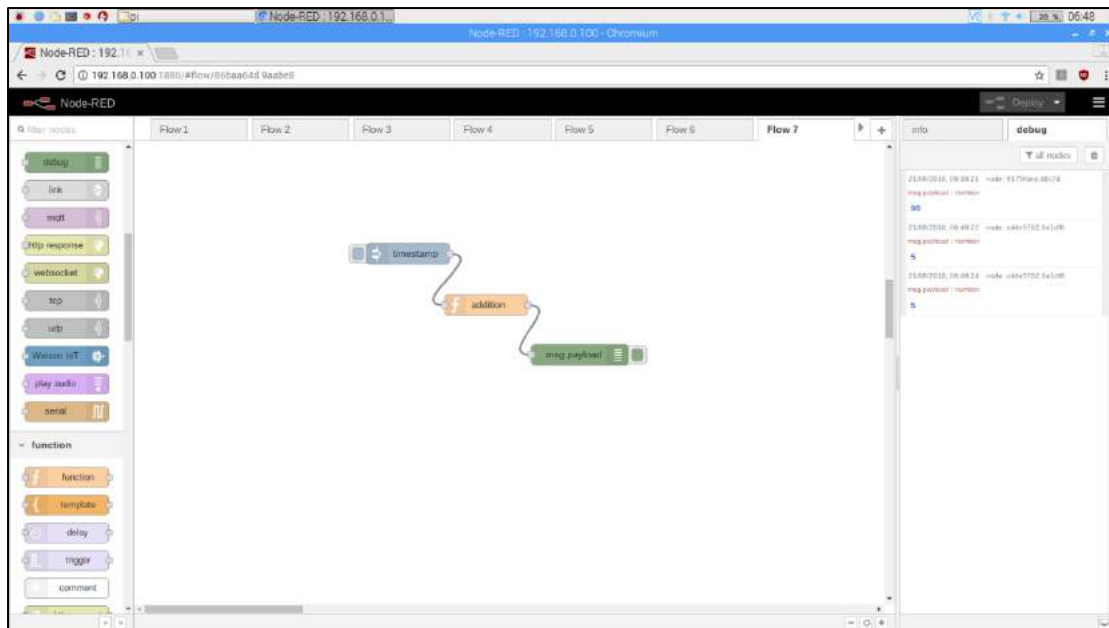
17. Function for adding two numbers. Create the flow as given below.



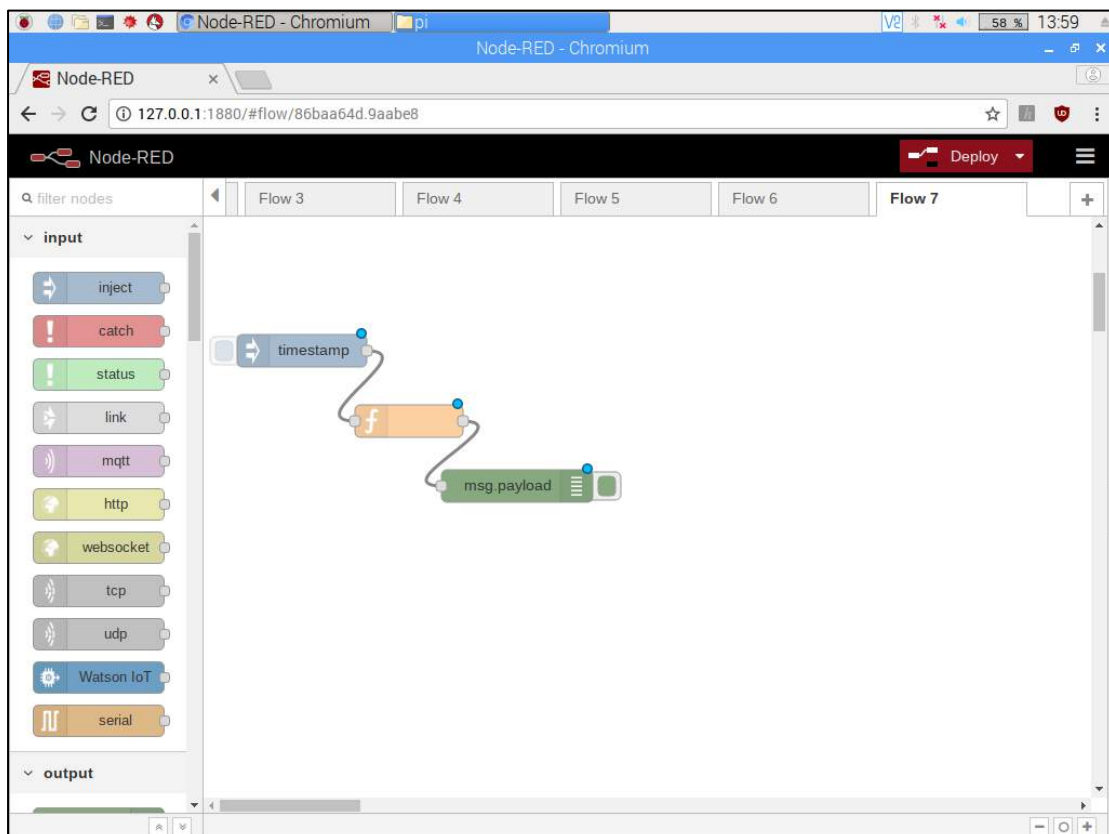
18. Double click the function and write the below given code



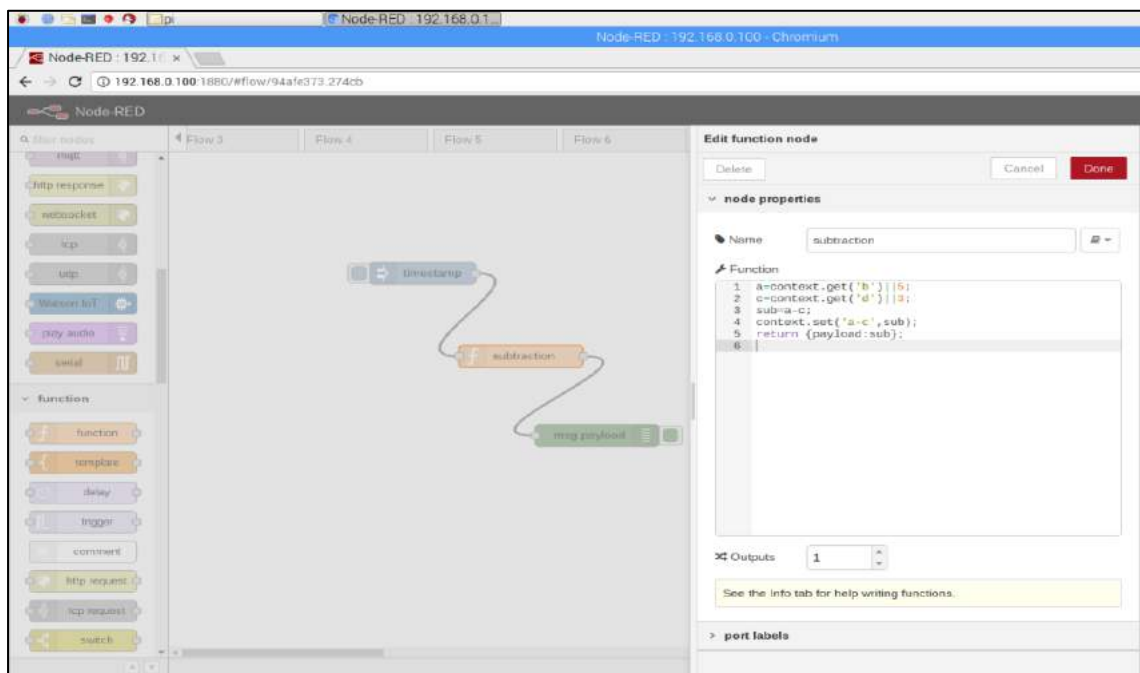
19. Deploy and check the output.



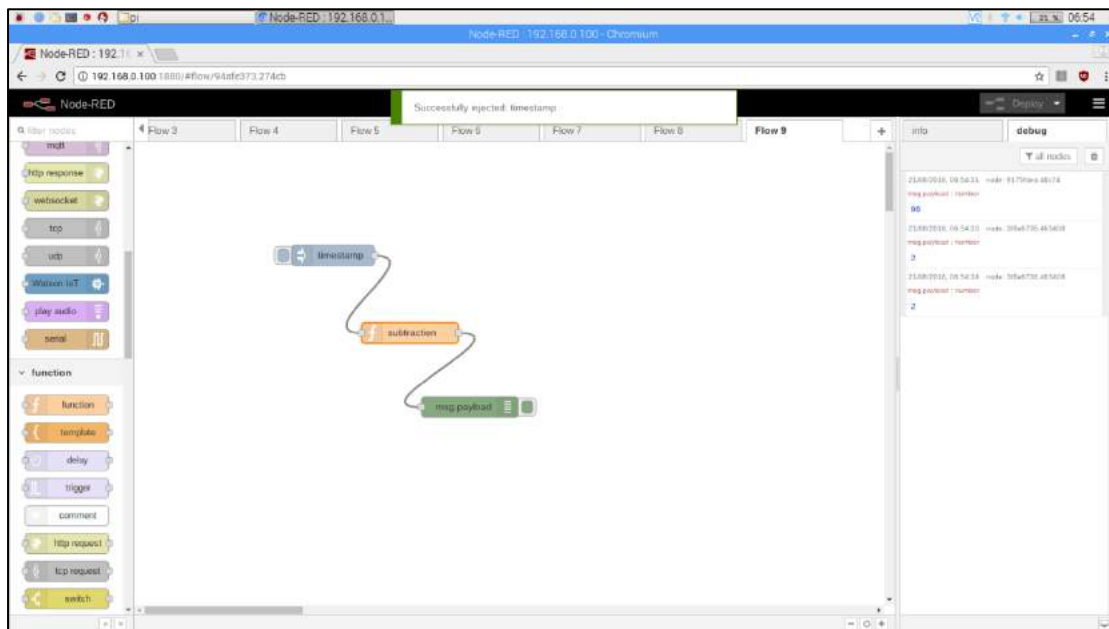
20. Function to subtract 2 numbers. Create the flow as given below.



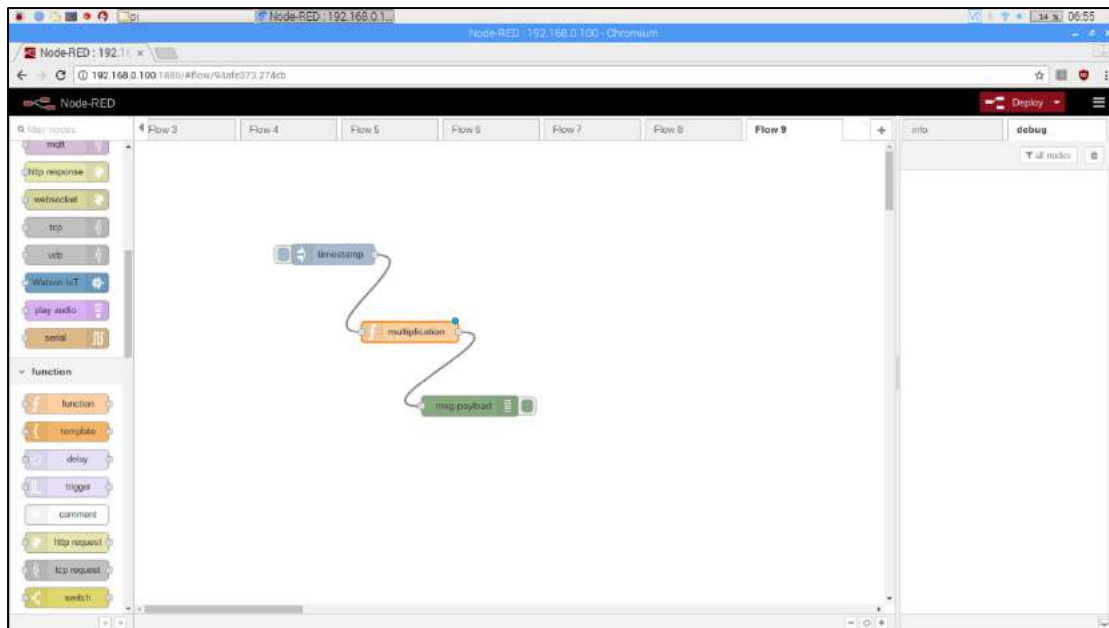
21. Double click the function and copy the code.



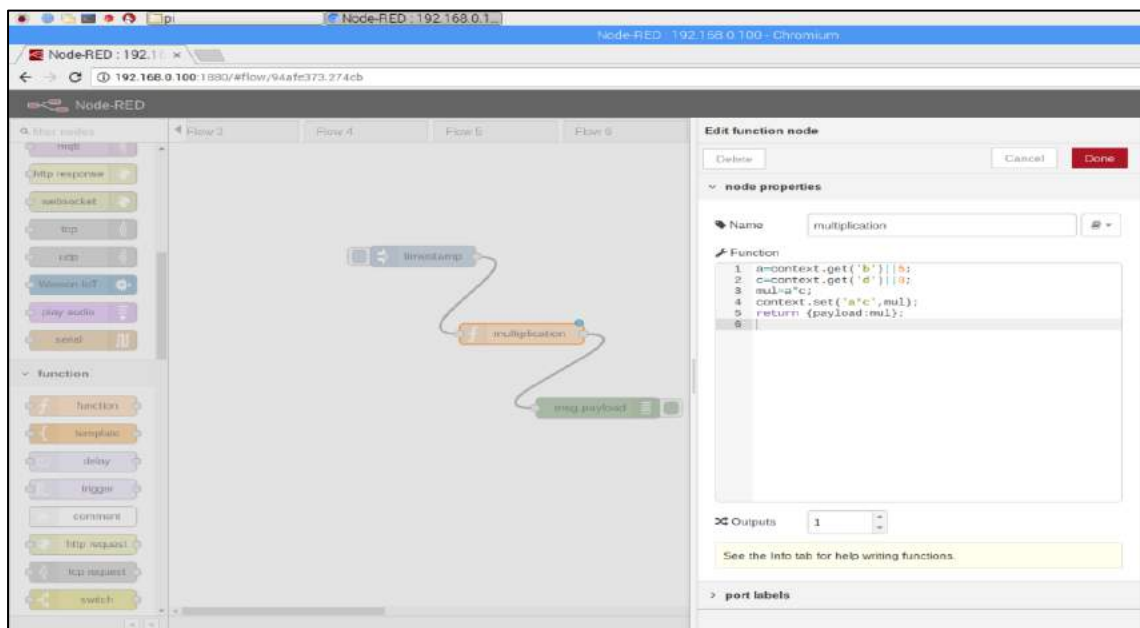
22. Deploy and check the output.



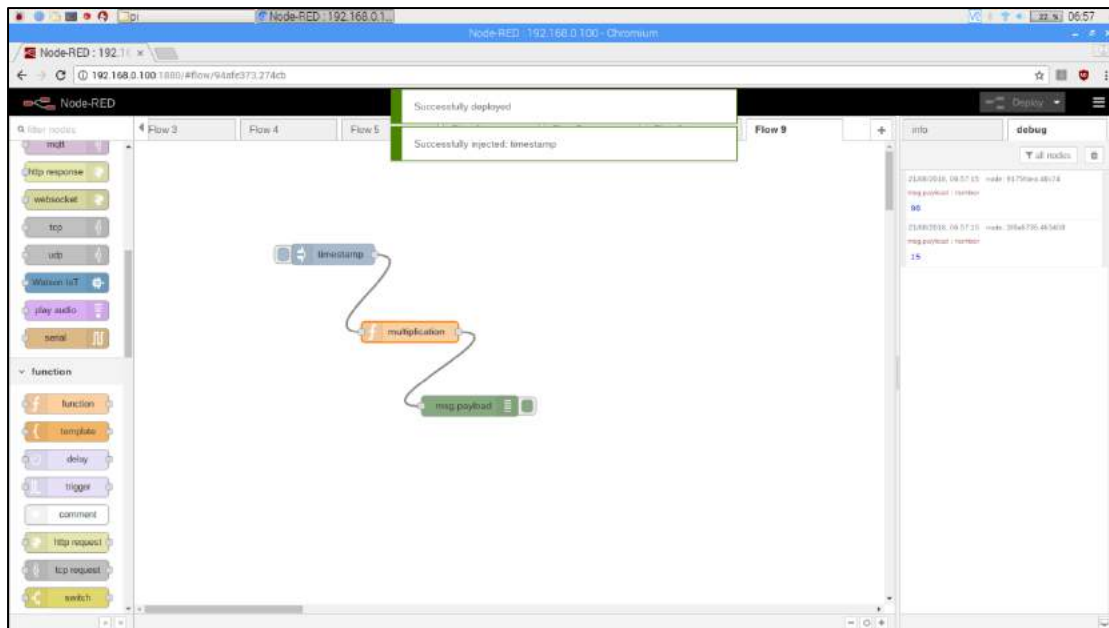
23. Function for multiplying 2 numbers. Create the flow as given below.



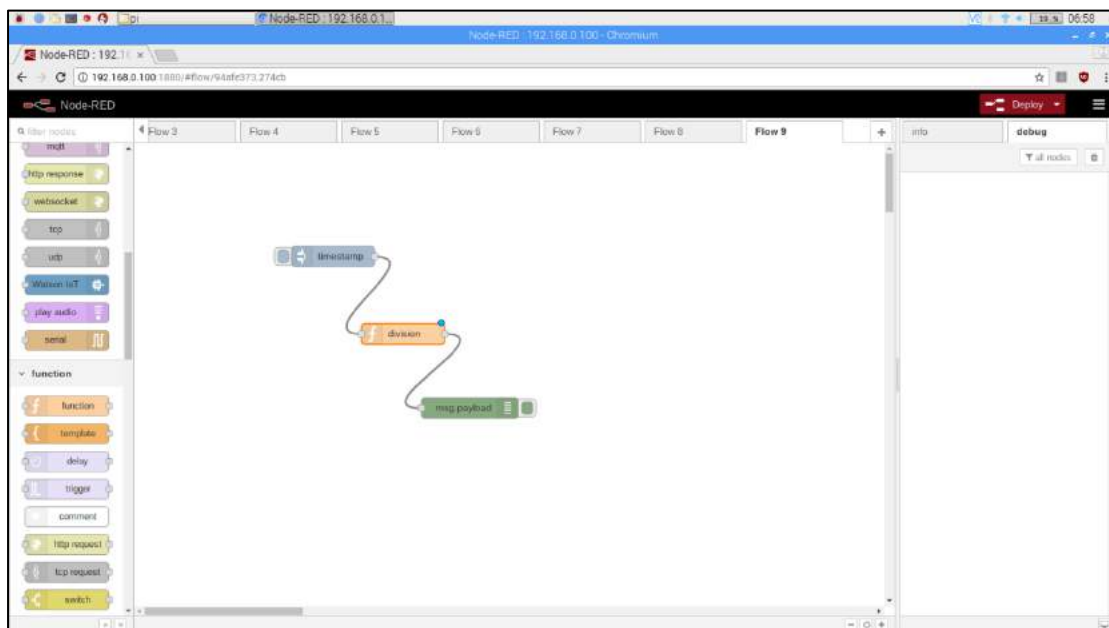
24. Double click the function and write the below code.



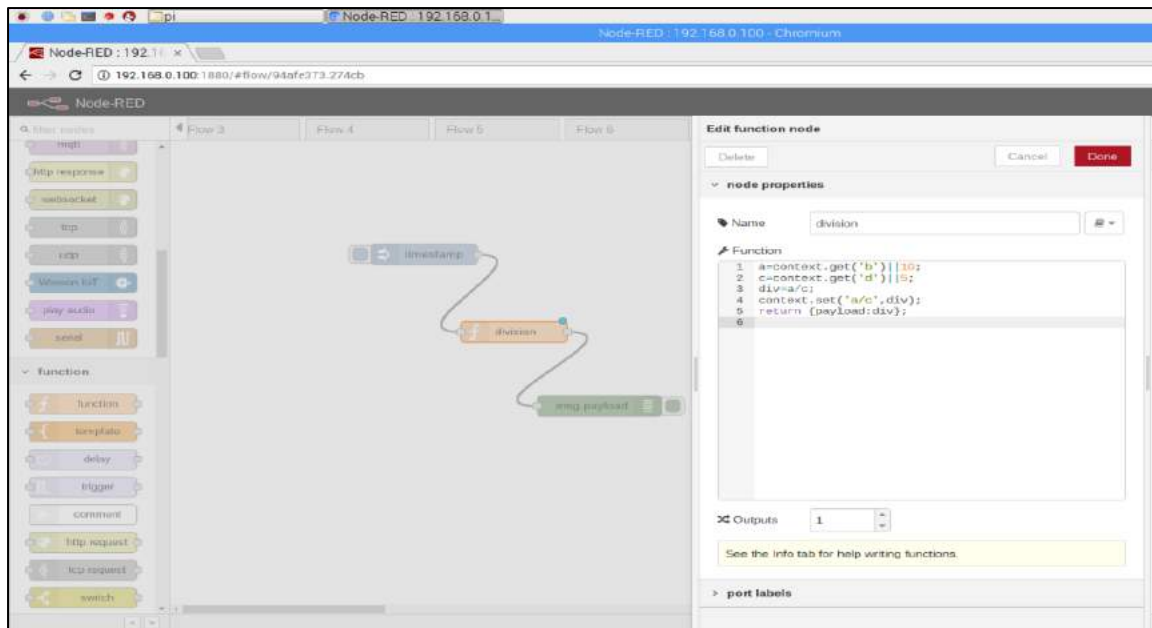
25. Deploy and check the result.



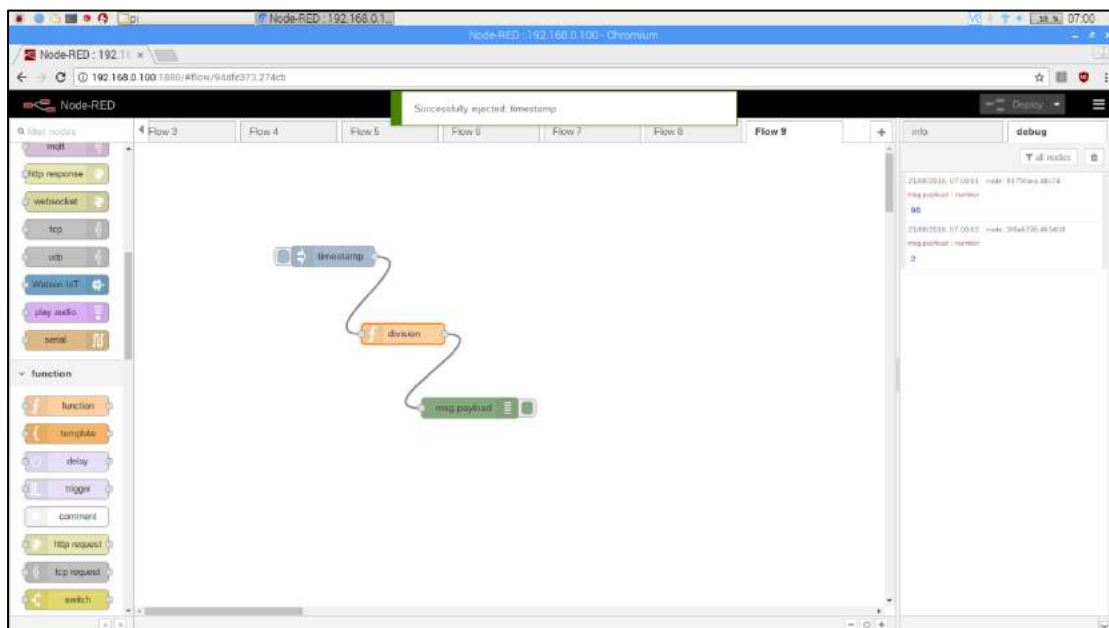
26. Function for dividing 2 numbers. Create the flow as given below.



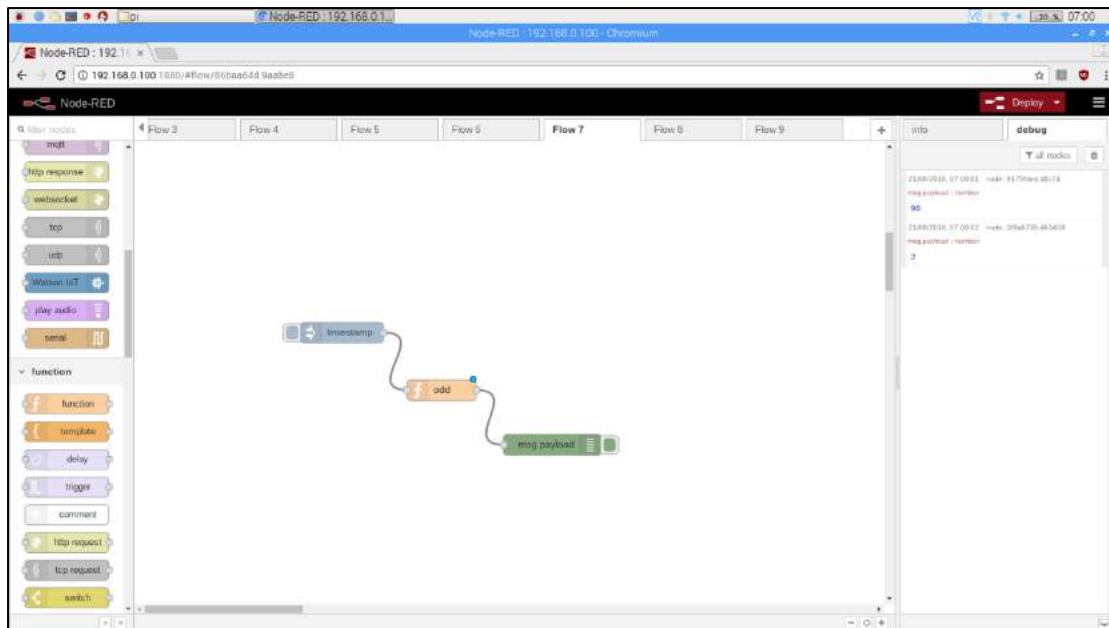
27. Double click the function and write the below code.



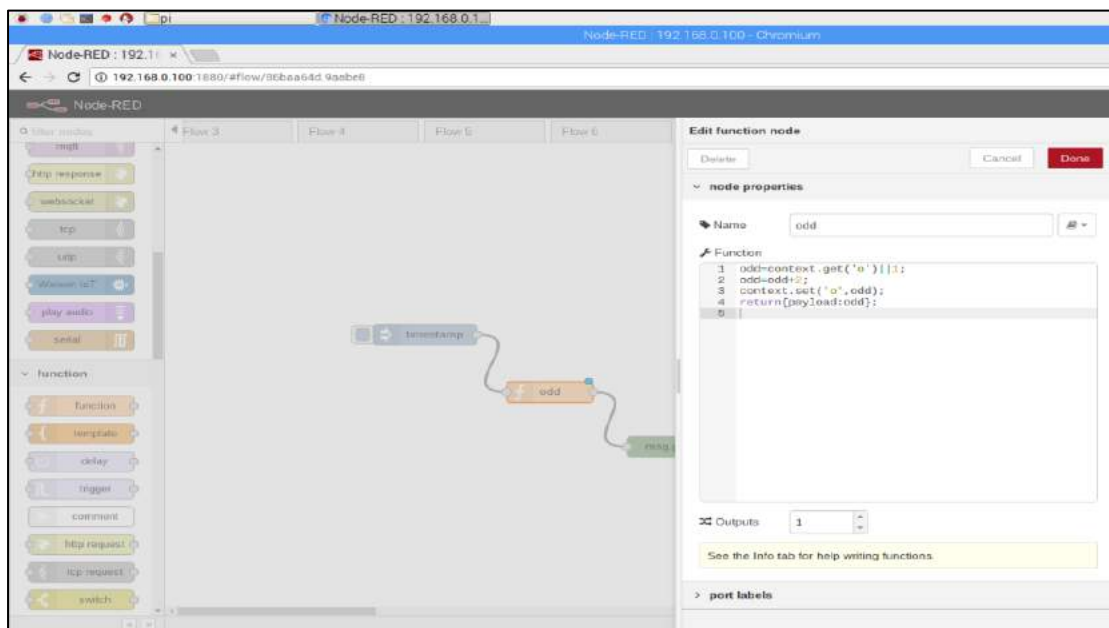
28. Deploy and check the output.



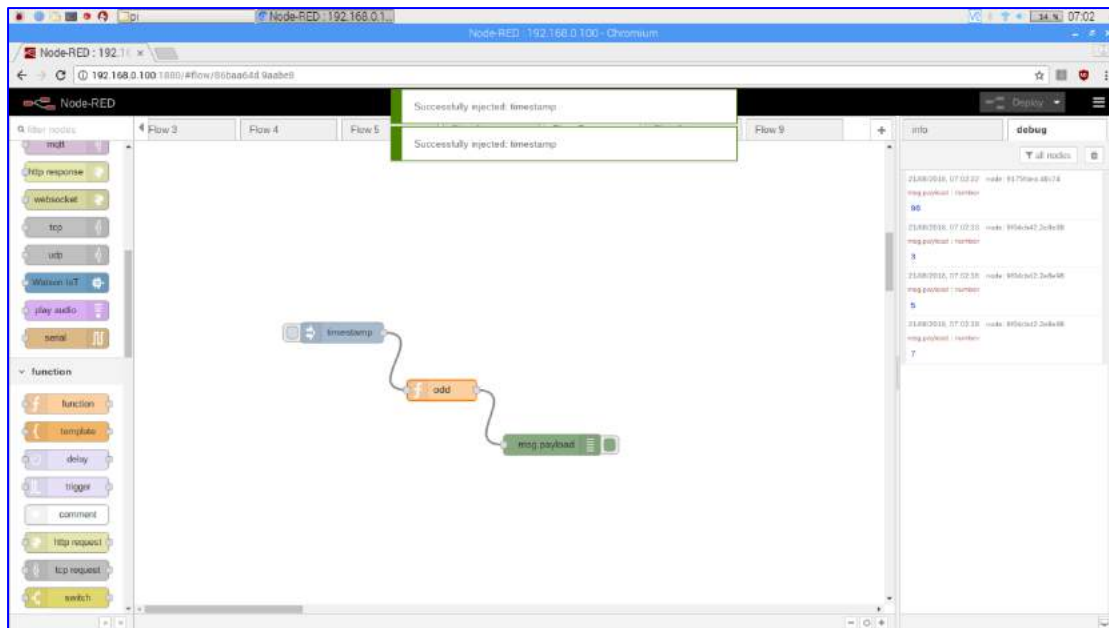
29. Function for generating odd numbers . Create the flow as given below.



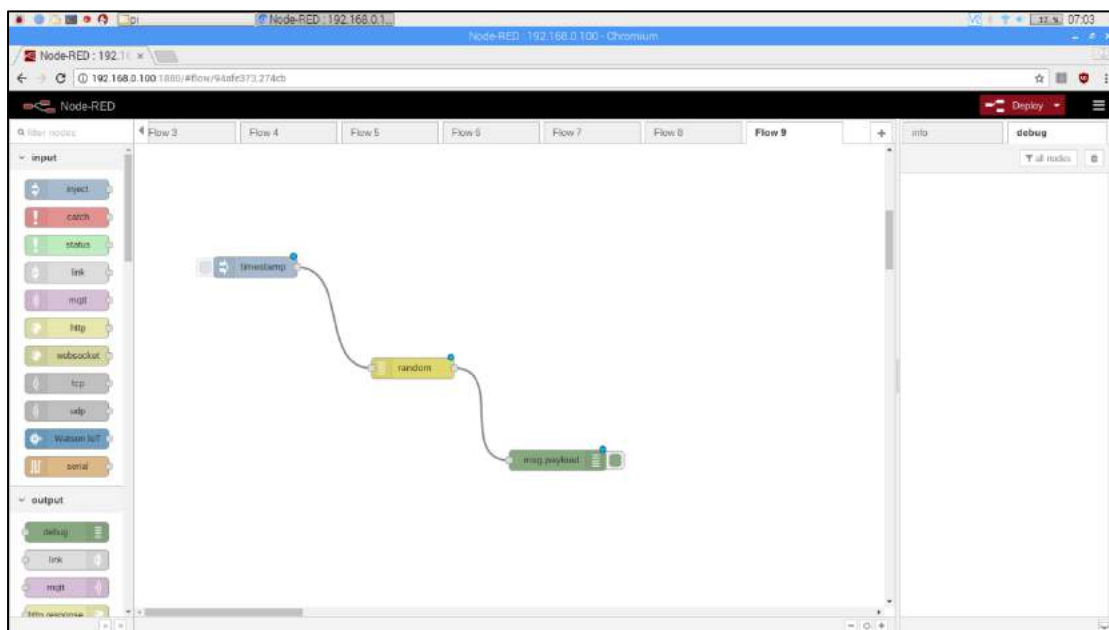
30. Double click the function and write the below code.



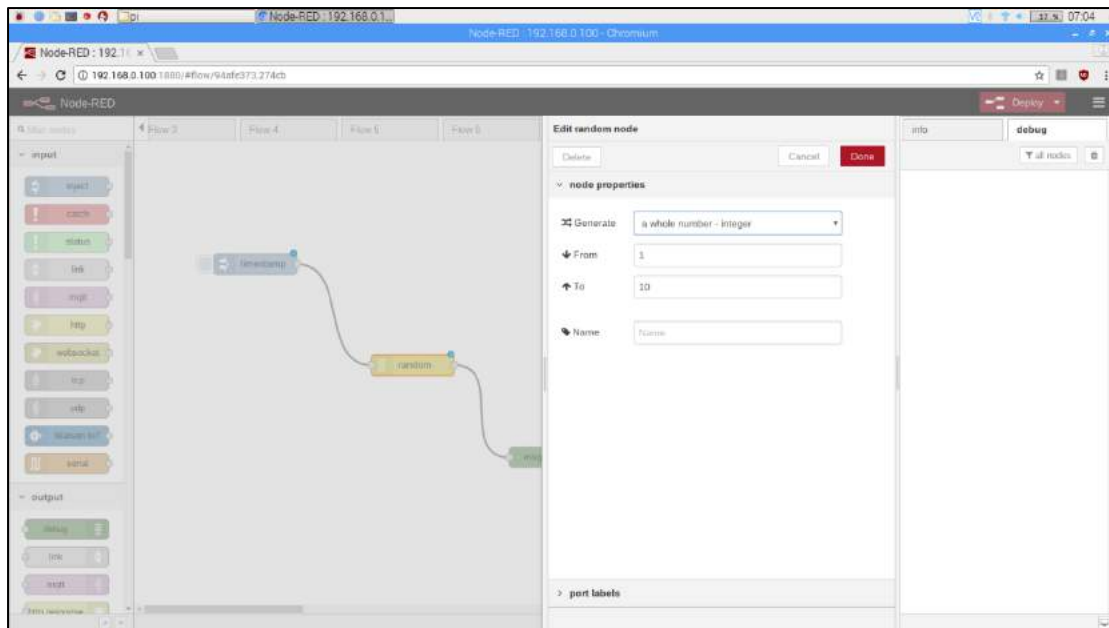
31. Deploy and check the result.



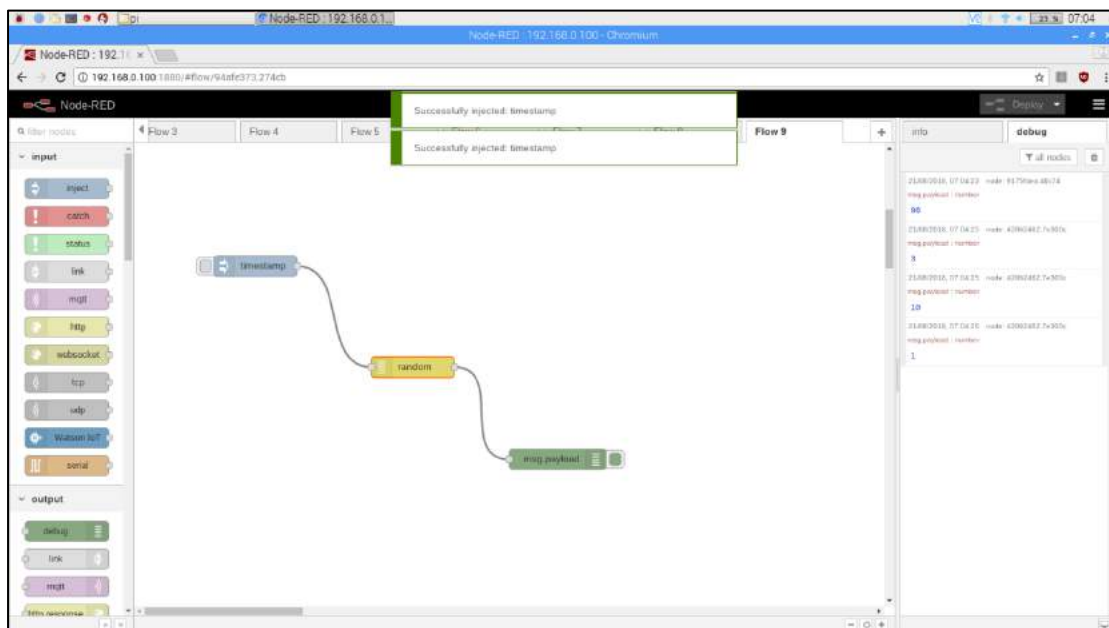
32. Function for random numbers. Create the flow given below by taking inbuilt random function.



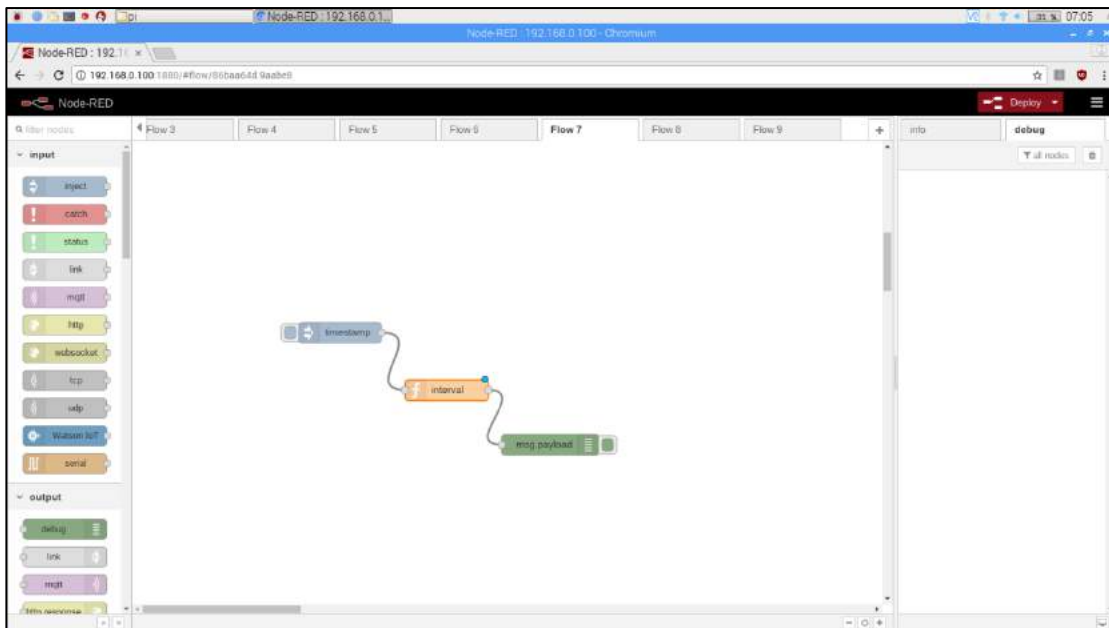
33. Double click the random function and make changes as given below.



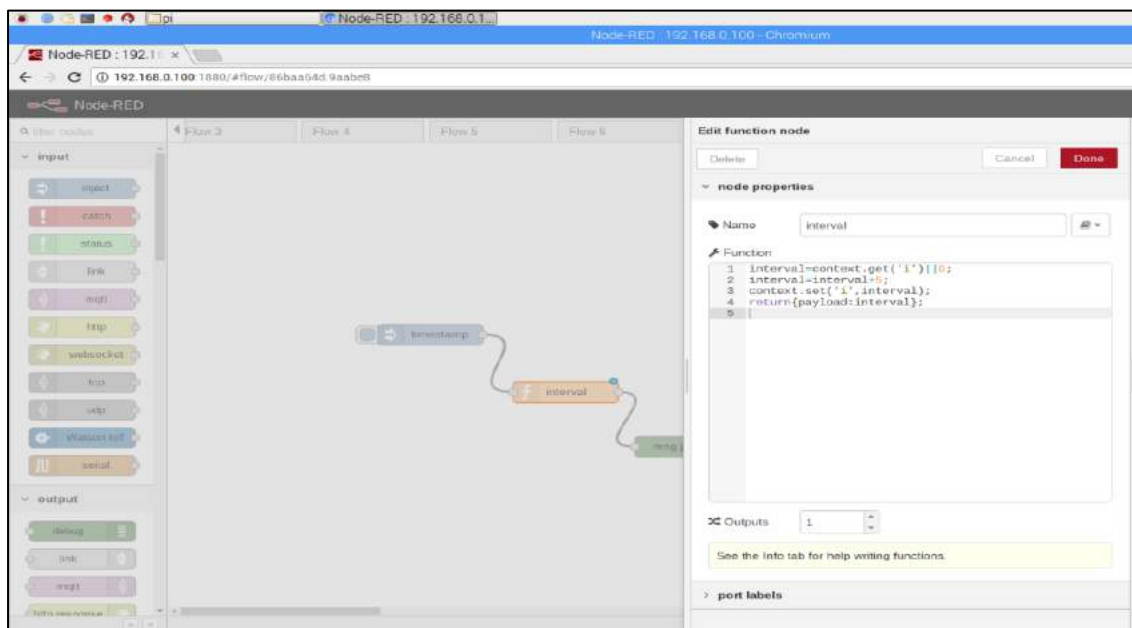
34. Deploy and check the output.



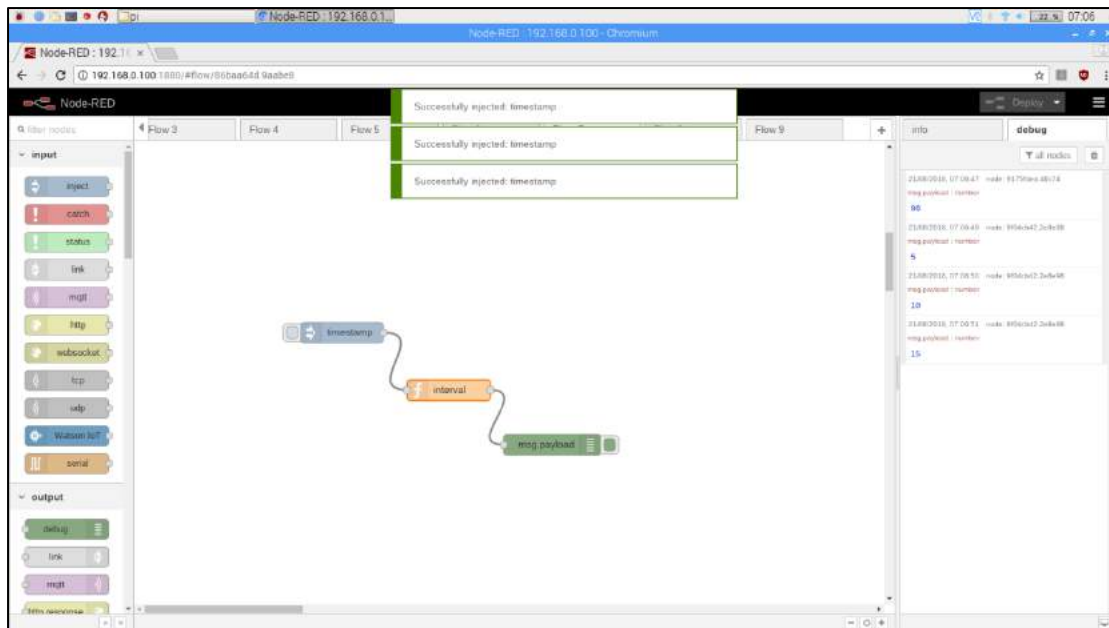
35. Function to get numbers with interval 5. create the flow as given below.



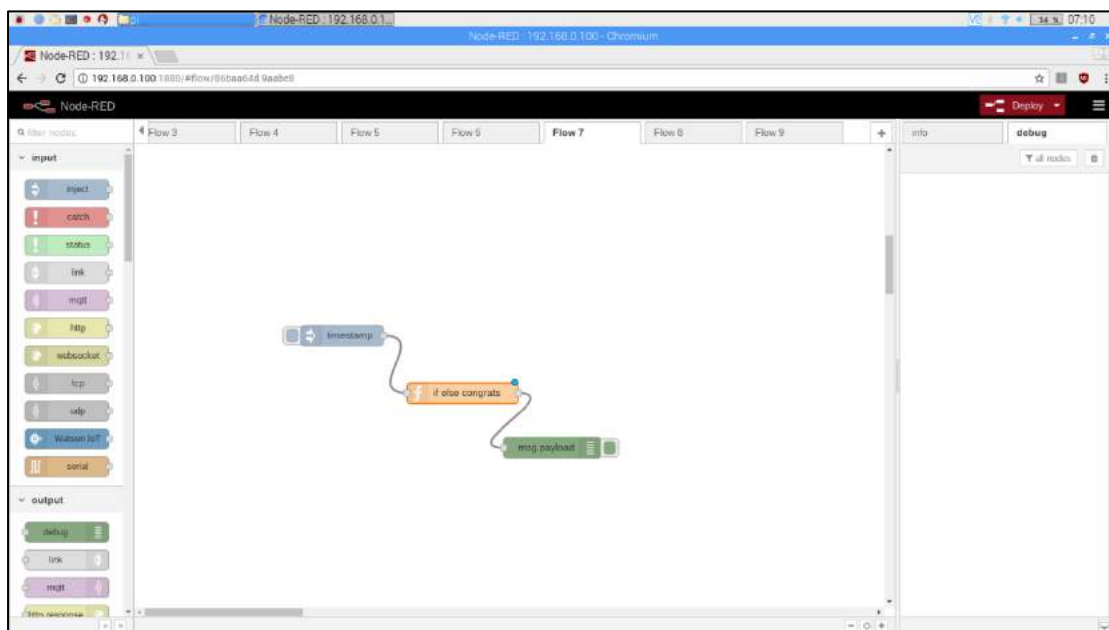
36. Double click the function node and write the code as below.



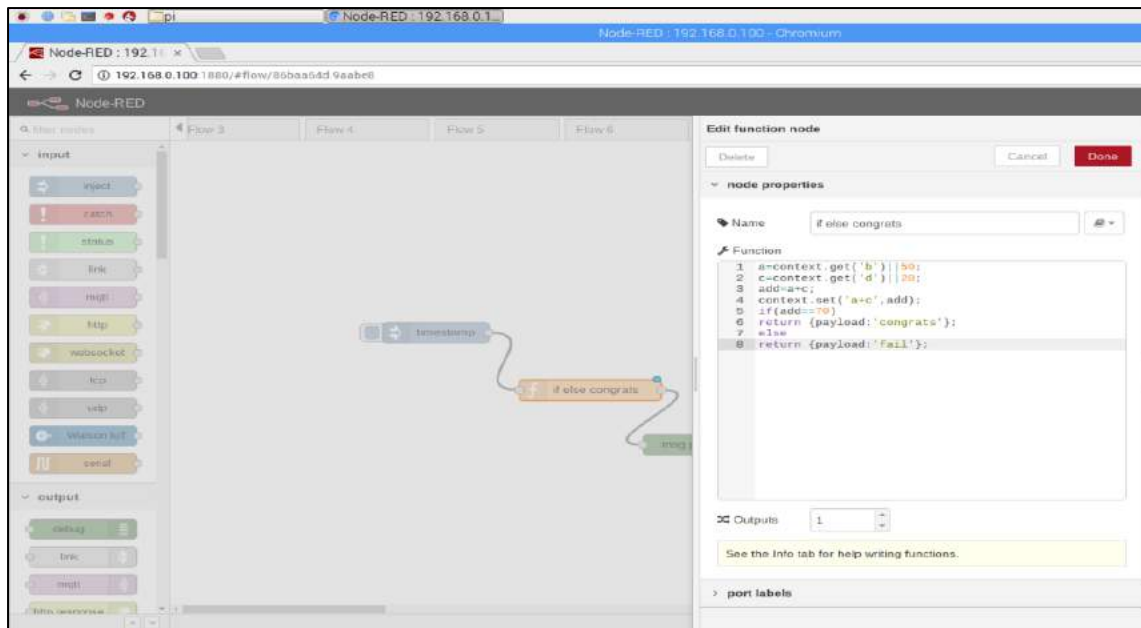
37. Deploy and check the output.



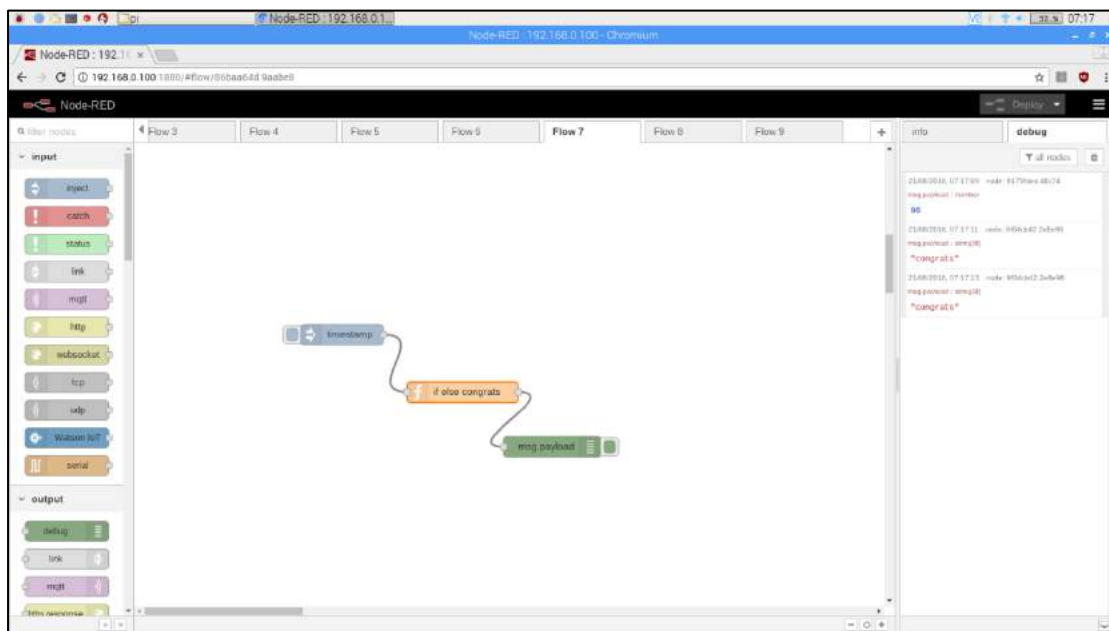
38. Function to print message “congrats” if marks are equal to 70. Create the flow as given below.



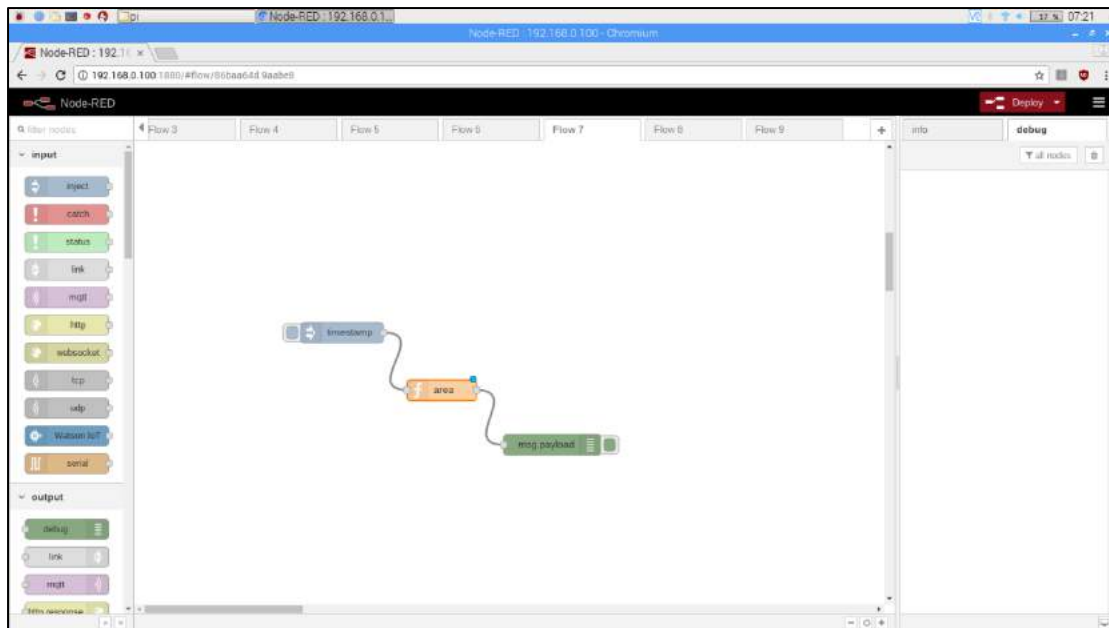
39. By double clicking the function Write the code as given below.



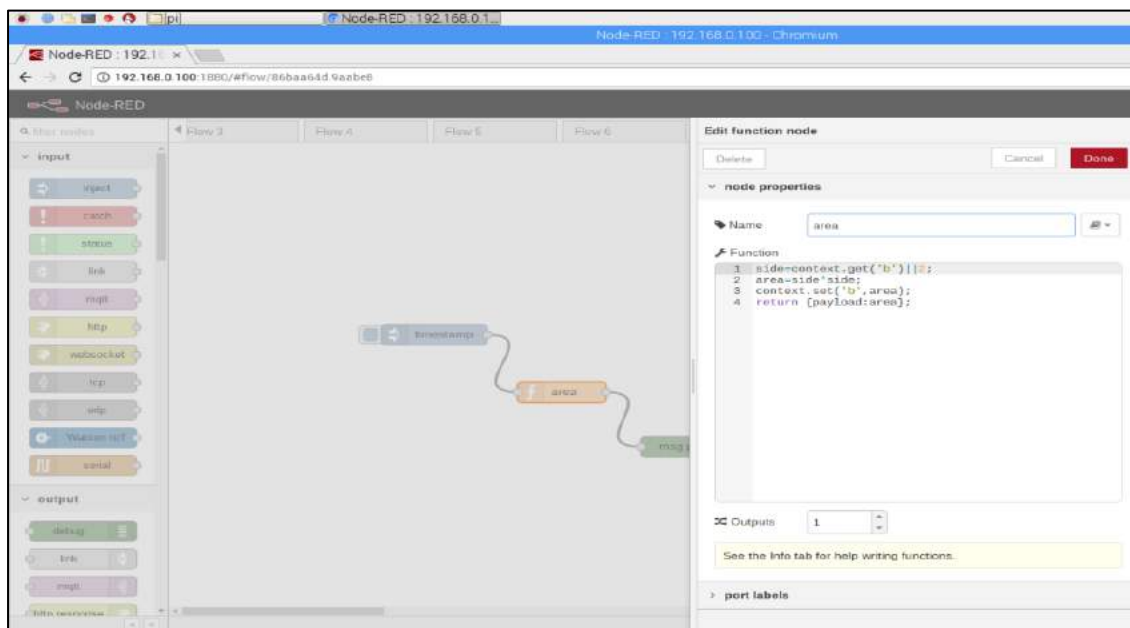
40. Deploy and check the output.



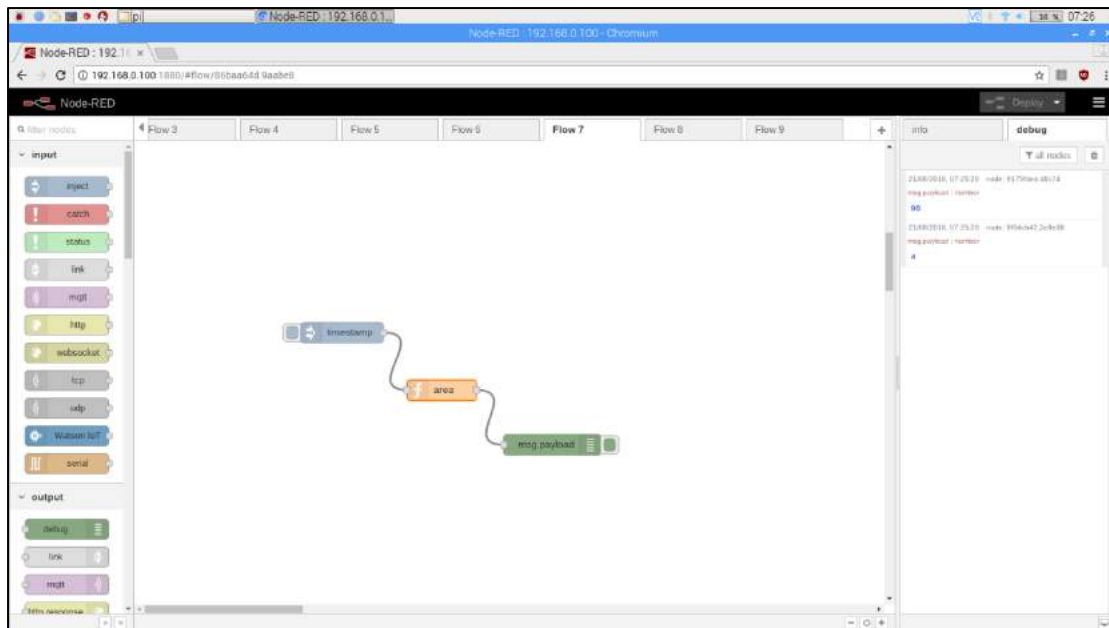
41. Function to calculate area of square. Create flow as given below



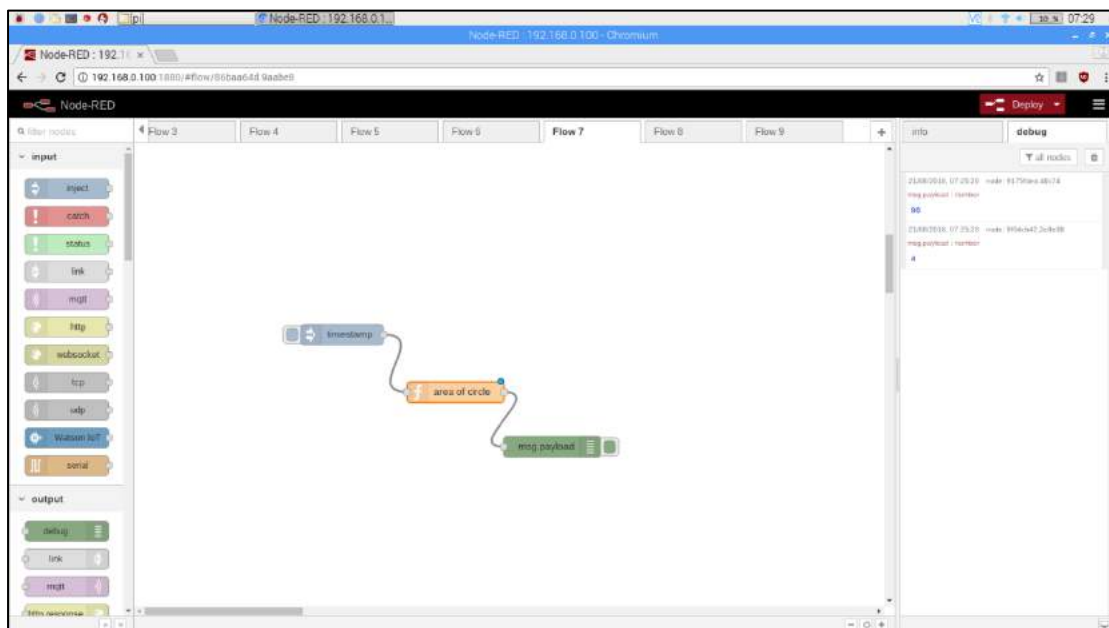
42. Double click the function and write the code.



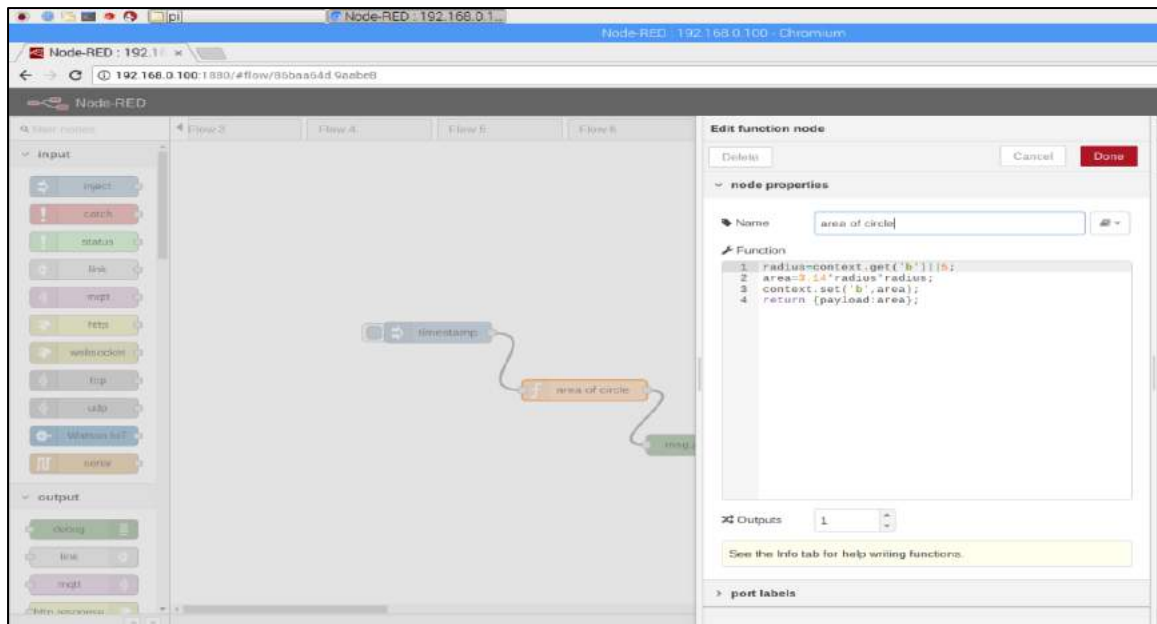
43. Deploy and check the result.



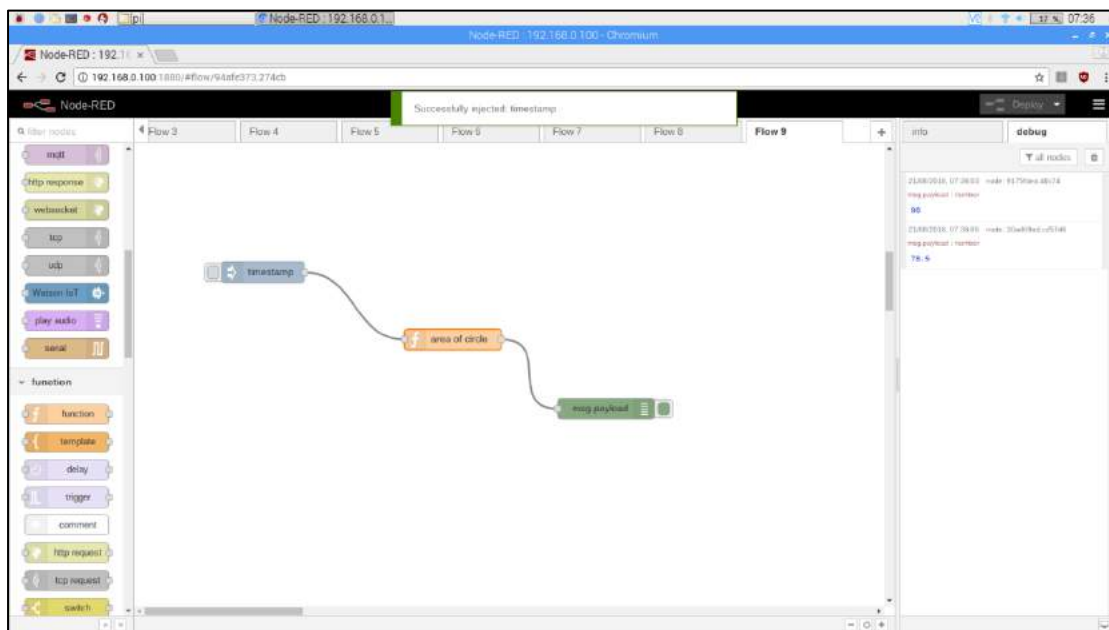
44. Function to calculate area of circle. Create the flow as given below.



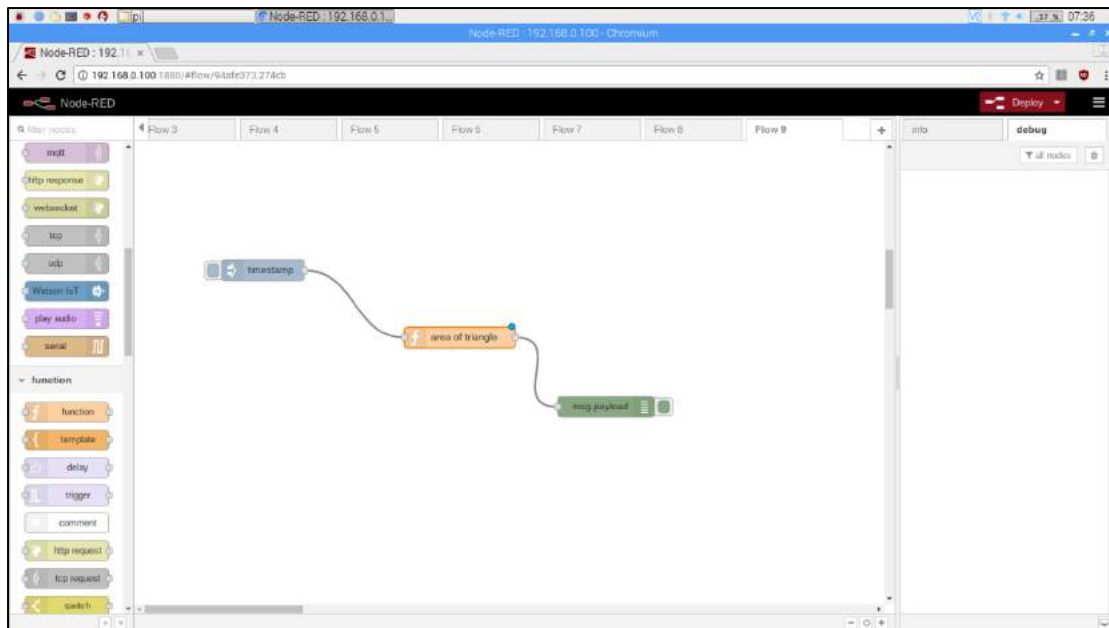
45. Double click the function and write the below code.



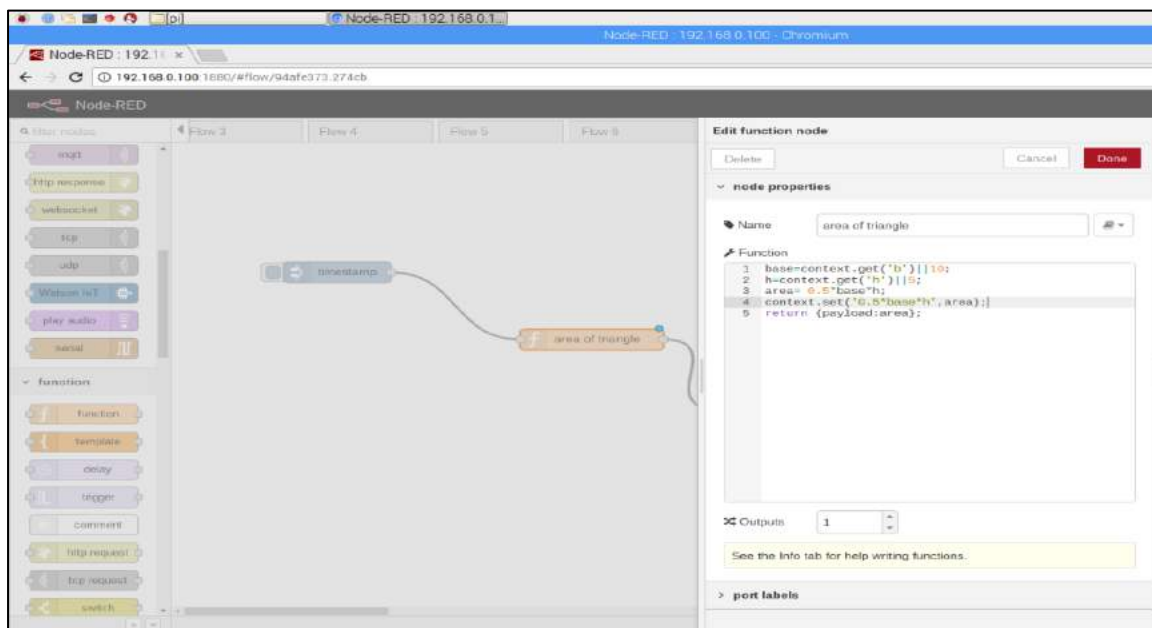
46. Deploy and check the output.



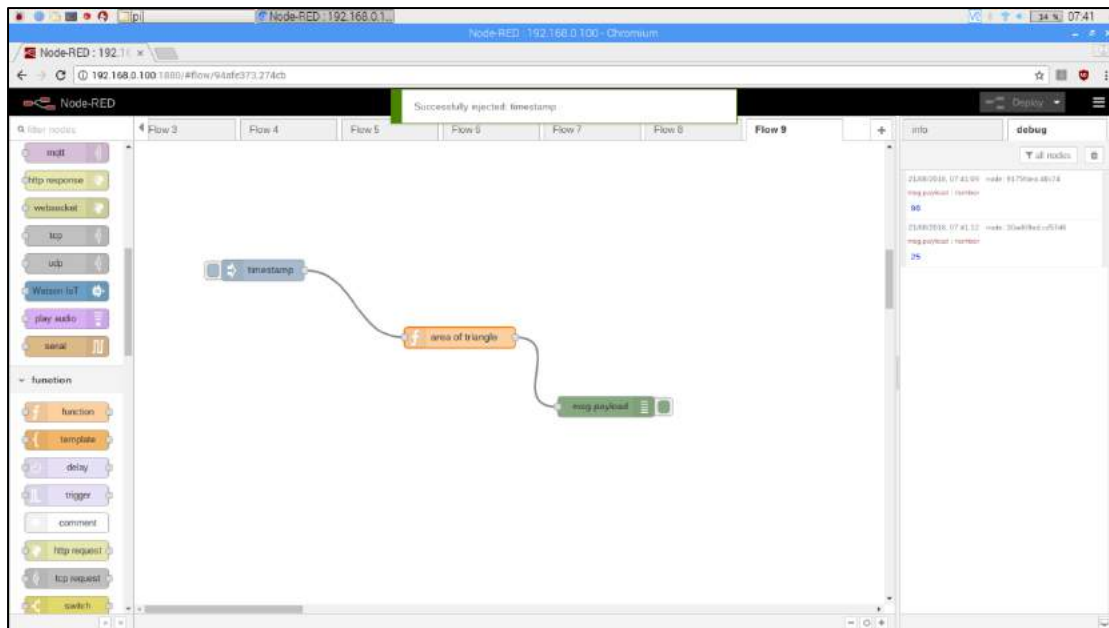
47. Function to calculate the area of triangle. Create the flow as given below.



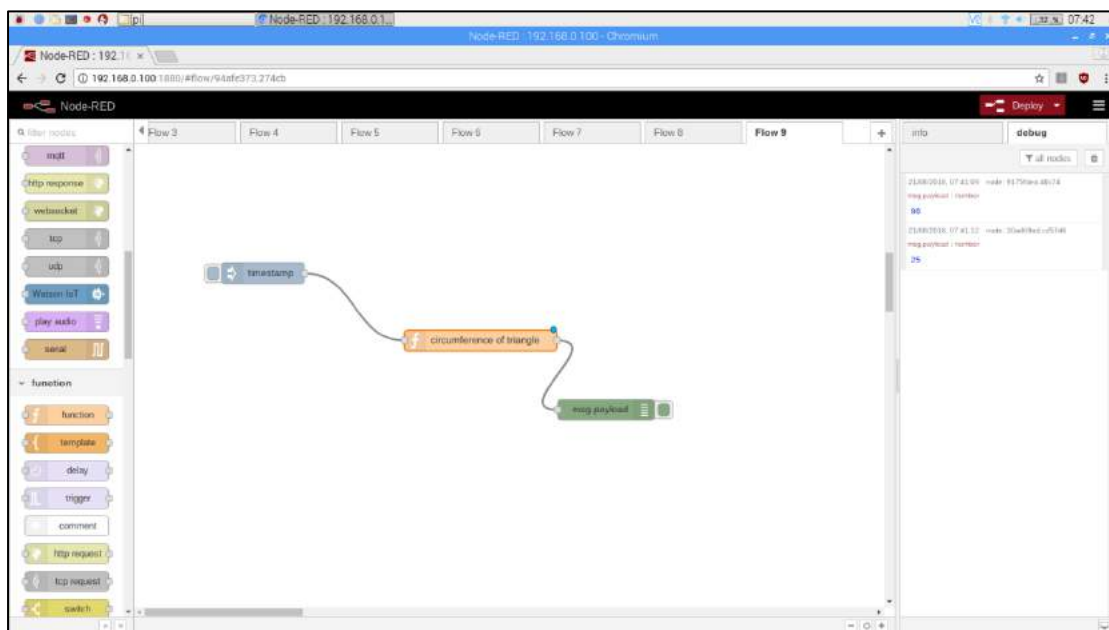
48. Double click the function and add the code .



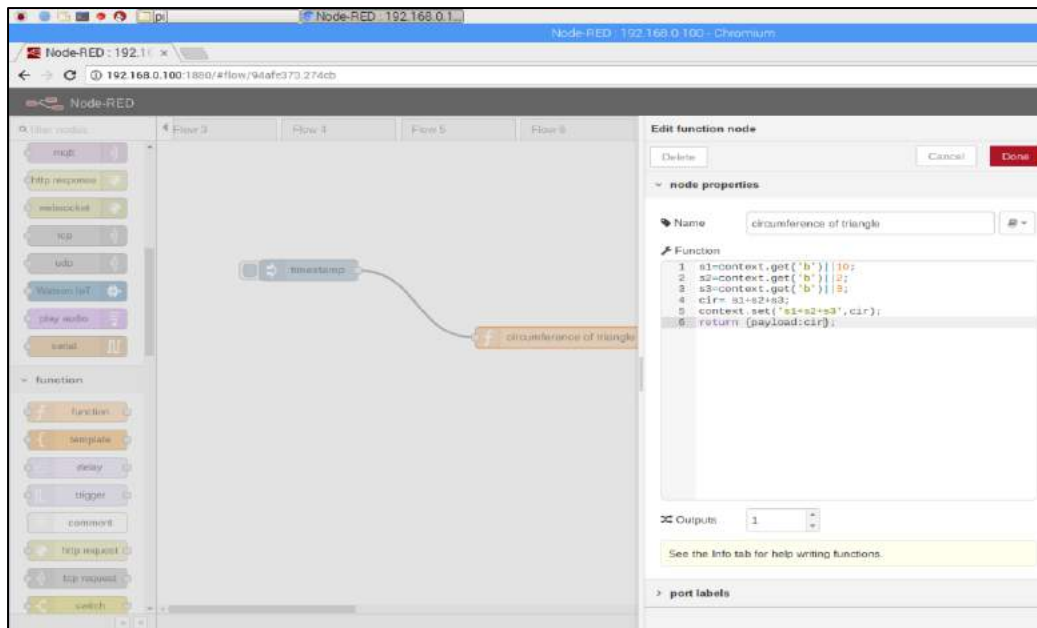
49. Deploy and check the result.



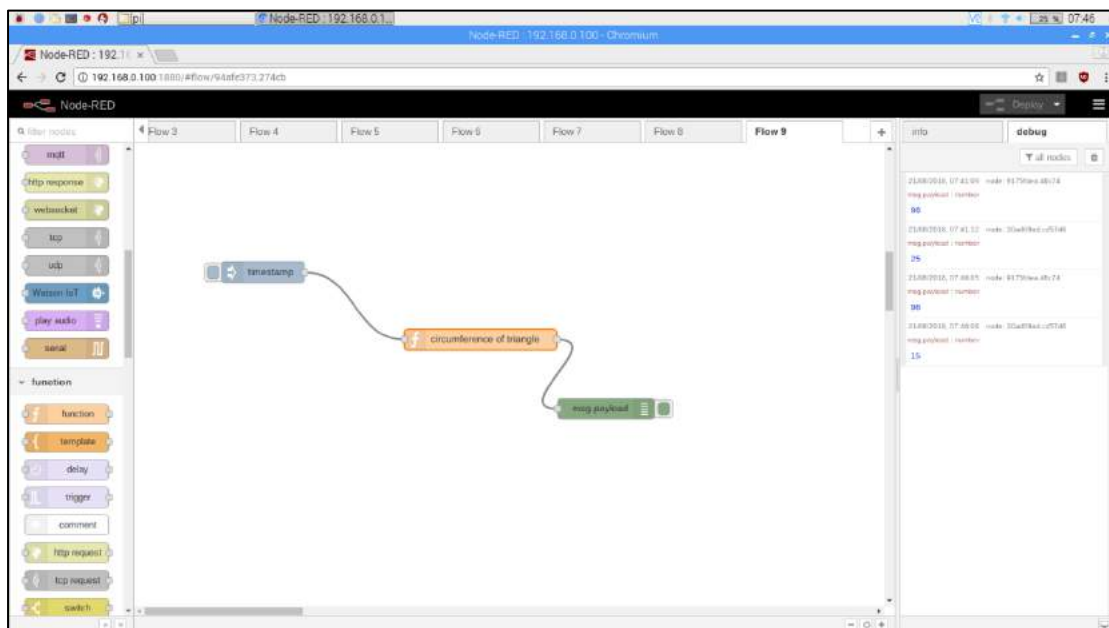
50. Function to calculate the perimeter of triangle. Create the flow as given below.



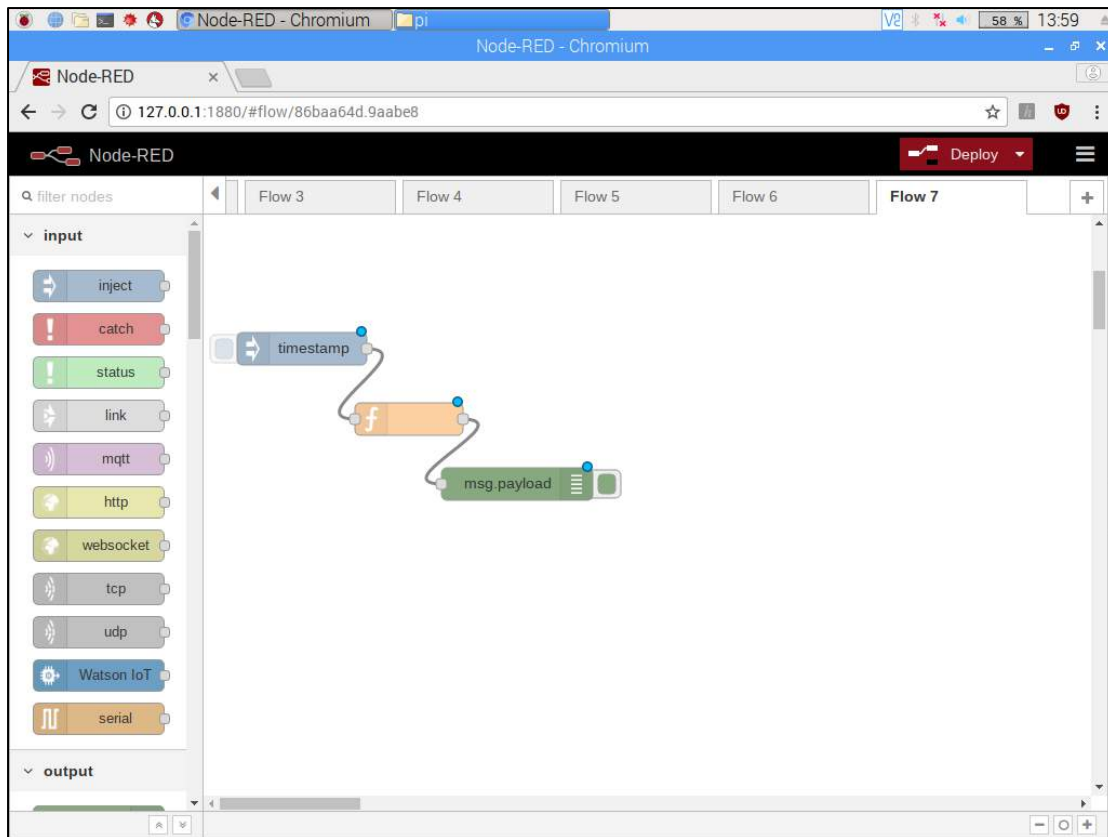
51. Double click the function and add the code given below.



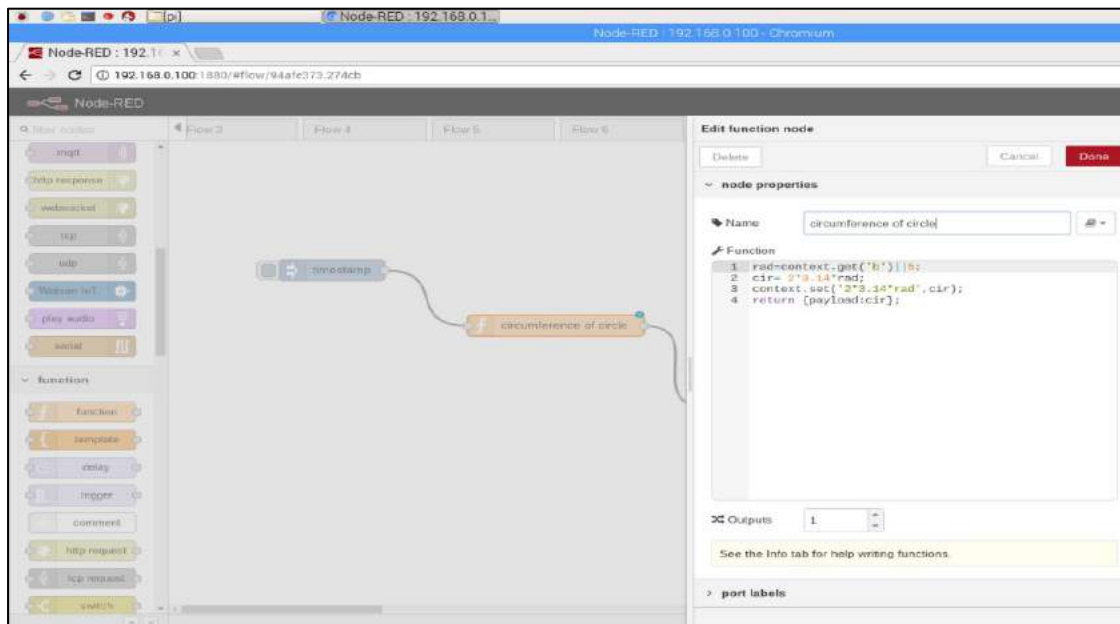
52. Deploy and check the output.



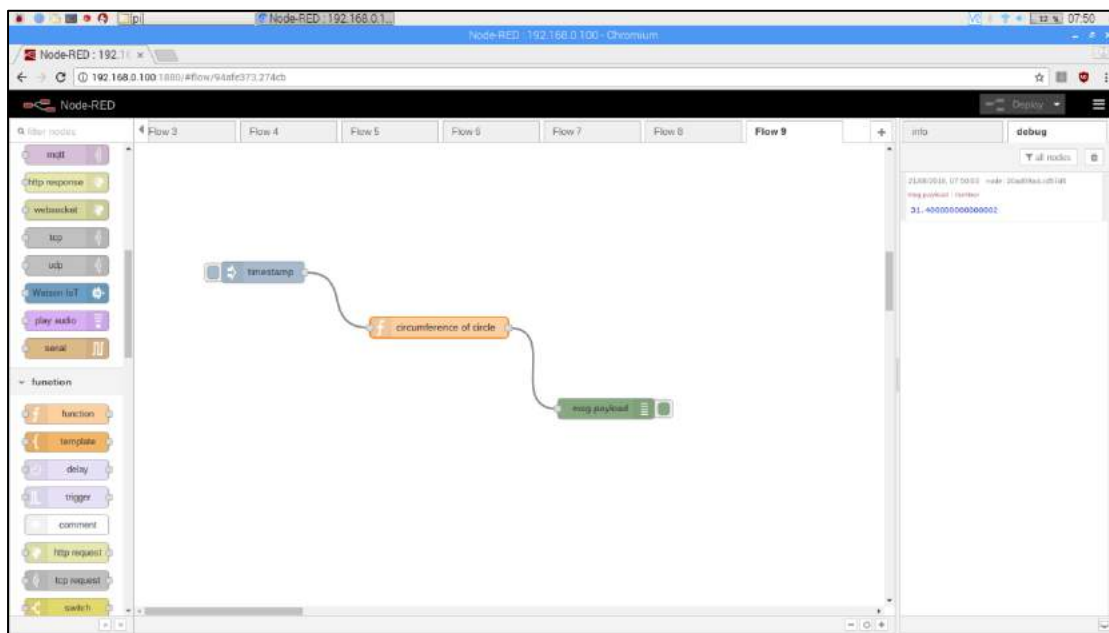
53. Function to calculate the circumference of circle. Create the flow as given below.



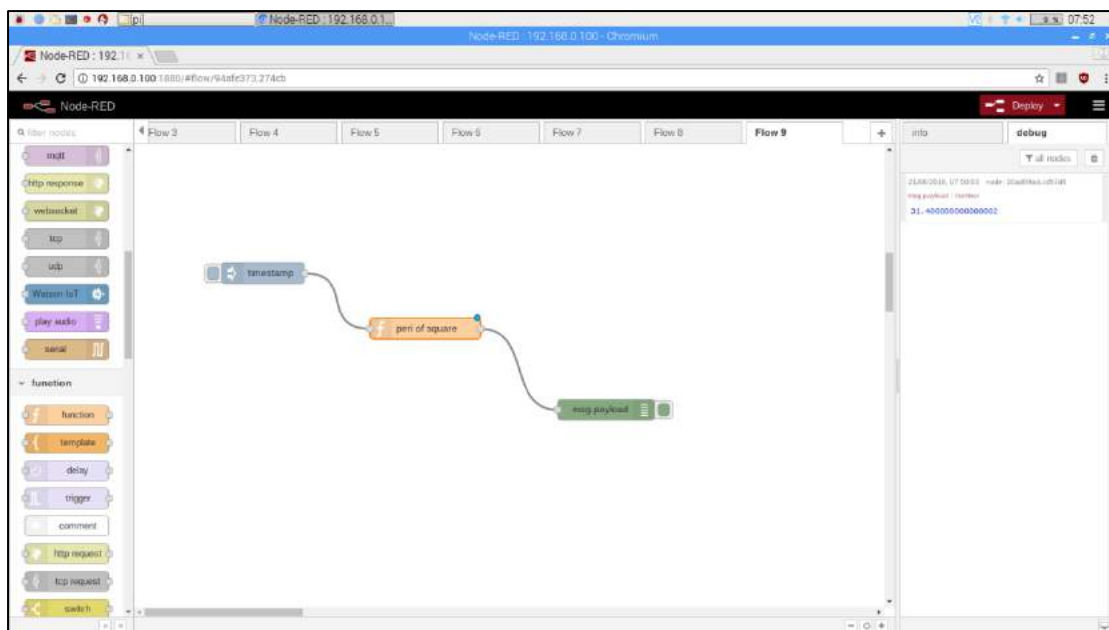
54. Double click the function and add the code.



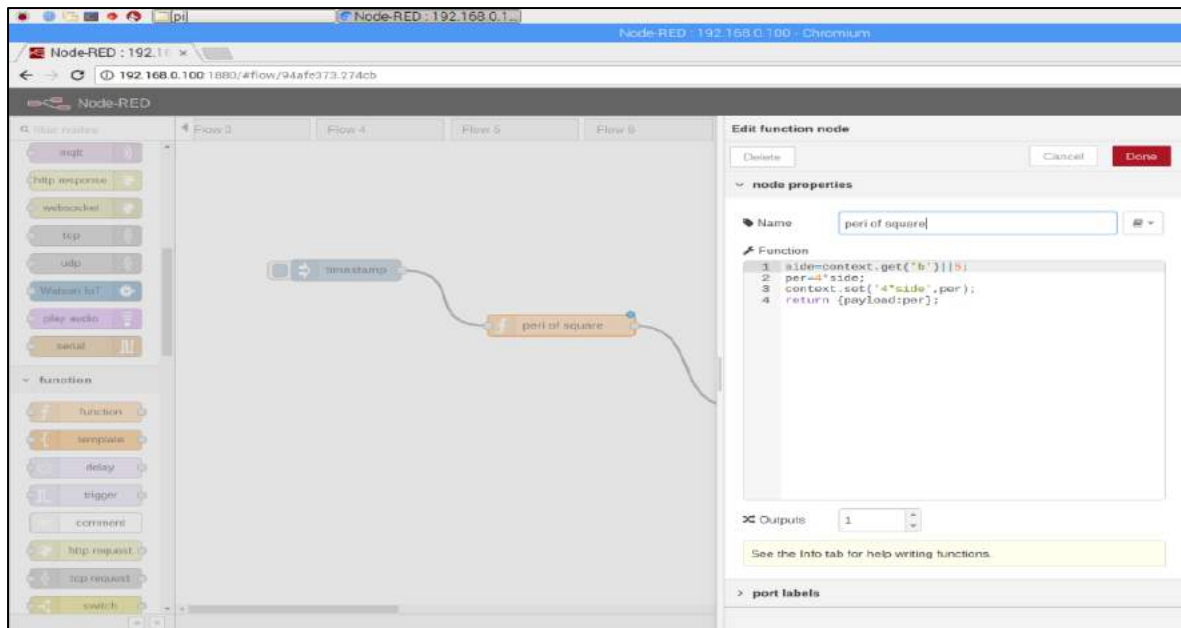
55. Deploy and check the output.



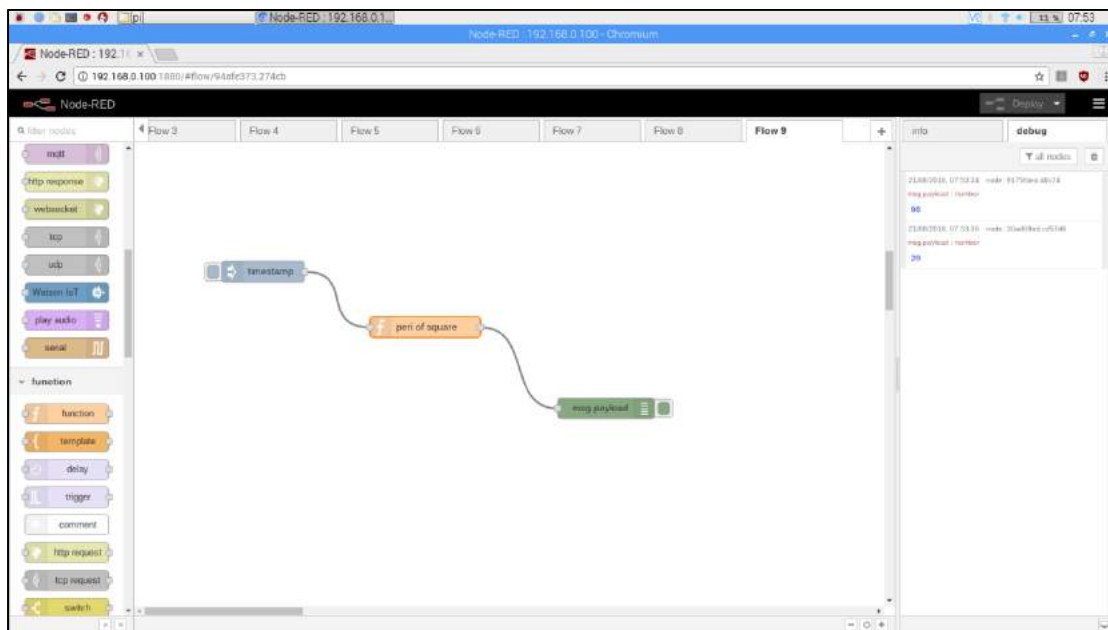
56. Function to calculate the perimeter of square. Create the flow as given below.



57. double click the function and add the code.



58. Deploy and check the output.



Practical 6:

Aim : How to install dashboard in NodeRED.

Step 1 : To Install NodeRED

Commands to Install NodeRED

Type the following commands in terminal

- `sudo apt-get update`
- `sudo apt-get upgrade`
- `sudo reboot`
- `sudo apt-get install npm`
- `npm -h`
- `sudo npm i -g npm@2x`

Step 2 : Go to node red and check “MANAGE PALLETE”

Step 3 : Go to install

Step 4 : Search dashboard and install

Step 5 : <http://192.168.0.110:1880/text> or page

Step 6 : <http://192.168.0.110:1880/ui>

Node-RED - Chromium

Node-RED - Chromium

127.0.0.1:1880/#flow/402a5021.f2d4e

Node-RED

Deploy

filter nodes

Flow 18 Flow 19 Flow 20

dashboard

- button
- dropdown
- switch
- slider
- numeric
- text input
- date picker
- colour picker
- form
- text
- gauge
- chart

CLICK ME

text abc

msg.payload

slider 41

gauge

chart

chart

Information

Flow	"402a5021.f2d4e"
Name	Flow 19
Status	Enabled

Flow Description

None

Import a flow by dragging its JSON into the editor, or with `ctrl-i`

Node-RED - Chromium

127.0.0.1:1880/#flow/402a5021.f2d4e

Node-RED

Deploy

filter nodes

Flow 16 Flow 17

dashboard

- button
- dropdown
- switch
- slider
- numeric
- text input
- date picker
- colour picker
- form
- text
- gauge
- chart

CLICK ME

slider 41

Edit button node

Delete Cancel Done

node properties

Group S1 [SUMMER]

Size auto

Icon optional icon

Label CLICK ME

Colour optional text/icon color

Background optional background color

When clicked, send:

Payload $\frac{a}{z}$ WELCOME TO MCC

Topic

node settings

Node-RED - Chromium

127.0.0.1:1880/#flow/402a5021.f2d4e

Node-RED

Flow 16 Flow 17

filter nodes

dashboard

- button
- dropdown
- switch
- slider
- numeric
- text input
- date picker
- colour picker
- form
- text
- gauge
- chart

CLICK ME

slider 41

Edit text node

Delete Cancel Done

node properties

Group S1 [SUMMER]

Size auto

Label text

Value format {{msg.payload}}

Layout

labelvalue	labelvalue	labelvalue
label value	label value	

Name

node settings

Node-RED - Chromium

127.0.0.1:1880/#flow/402a5021.f2d4e

Node-RED

Deploy

filter nodes

Flow 16 Flow 17

dashboard

- button
- dropdown
- switch
- slider
- numeric
- text input
- date picker
- colour picker
- form
- text
- gauge
- chart

CLICK ME

slider

41

Edit debug node

Delete Cancel Done

node properties

Output msg.payload

To ☒ debug window ☐ system console ☐ node status (32 characters)

Name Name

node settings

Node-RED - Chromium

127.0.0.1:1880/#flow/402a5021.f2d4e

Node-RED

Flow 16 Flow 17

dashboard

- button
- dropdown
- switch
- slider
- numeric
- text input
- date picker
- colour picker
- form
- text
- gauge
- chart

CLICK ME

slider

41

Edit slider node

Delete Cancel Done

node properties

Group S1 [SUMMER]

Size auto

Label slider

Range min 0 max 200 step 1

→ If msg arrives on input, pass through to output: ☒

When changed, send:

Payload Current value

Topic

Name

node settings

Node-RED - Chromium

127.0.0.1:1880/#flow/402a5021.f2d4e

Node-RED

Deploy

filter nodes

Flow 16

Flow 17

dashboard

- button
- dropdown
- switch
- slider
- numeric
- text input
- date picker
- colour picker
- form
- text
- gauge
- chart

CLICK ME

slider

41

Edit gauge node

Delete Cancel Done

node properties

Group S1 [SUMMER]

Size auto

Type Gauge

Label gauge

Value format {{value}}

Units units

Range min 0 max 200

Colour gradient

Sectors 0 70 100 200

node settings

Node-RED - Chromium

127.0.0.1:1880/#flow/402a5021.f2d4e

Node-RED

Flow 16 Flow 17

dashboard

- button
- dropdown
- switch
- slider
- numeric
- text input
- date picker
- colour picker
- form
- text
- gauge
- chart

CLICK ME

slider 41

Edit chart node

Delete Cancel Done

node properties

Group S1 [SUMMER]

Size auto

Label chart

Type Line chart ☐ enlarge points

X-axis last 1 hours OR 1000 points

X-axis Label HH:mm:ss

Y-axis min 0 max 200

Legend None Interpolate linear

Series Colours

node settings

Node-RED - Chromium

127.0.0.1:1880/#flow/402a5021.f2d4e

Node-RED

Deploy

filter nodes

Flow 16

Flow 17

dashboard

- button
- dropdown
- switch
- slider
- numeric
- text input
- date picker
- colour picker
- form
- text
- gauge
- chart

CLICK ME

slider

41

Edit chart node

Delete Cancel Done

node properties

Group S1 [SUMMER]

Size auto

Label chart

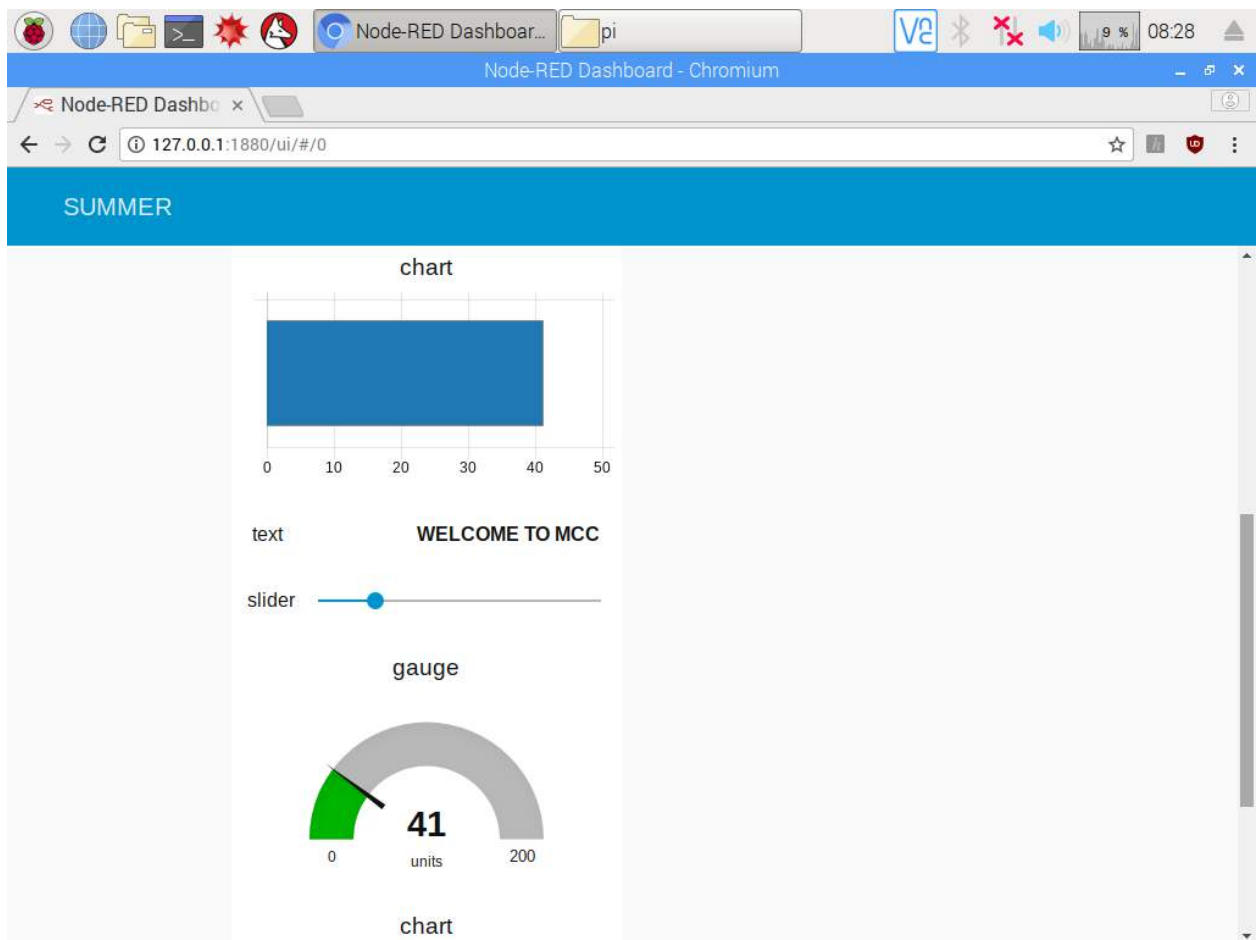
Type Bar chart (H)

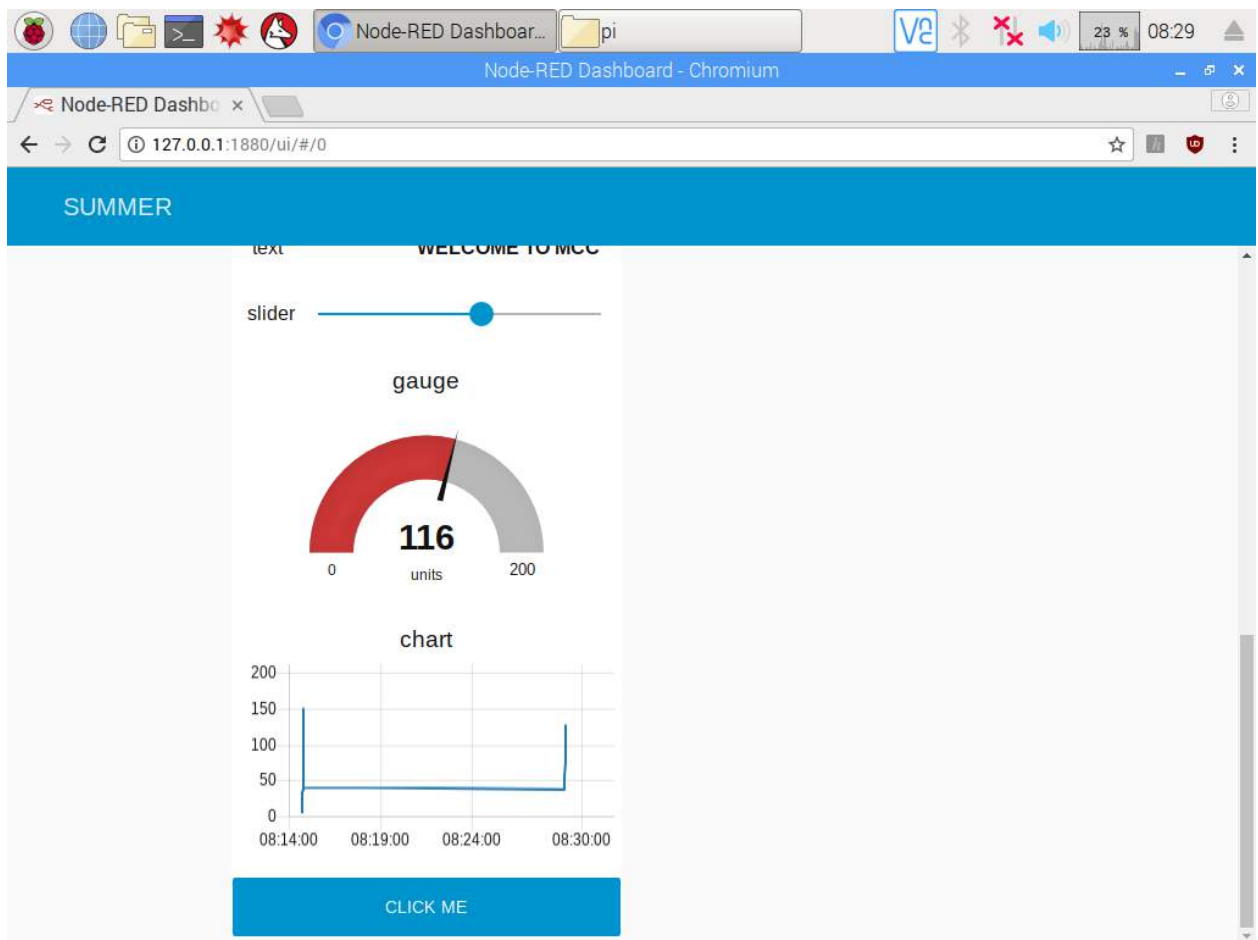
X-axis min max

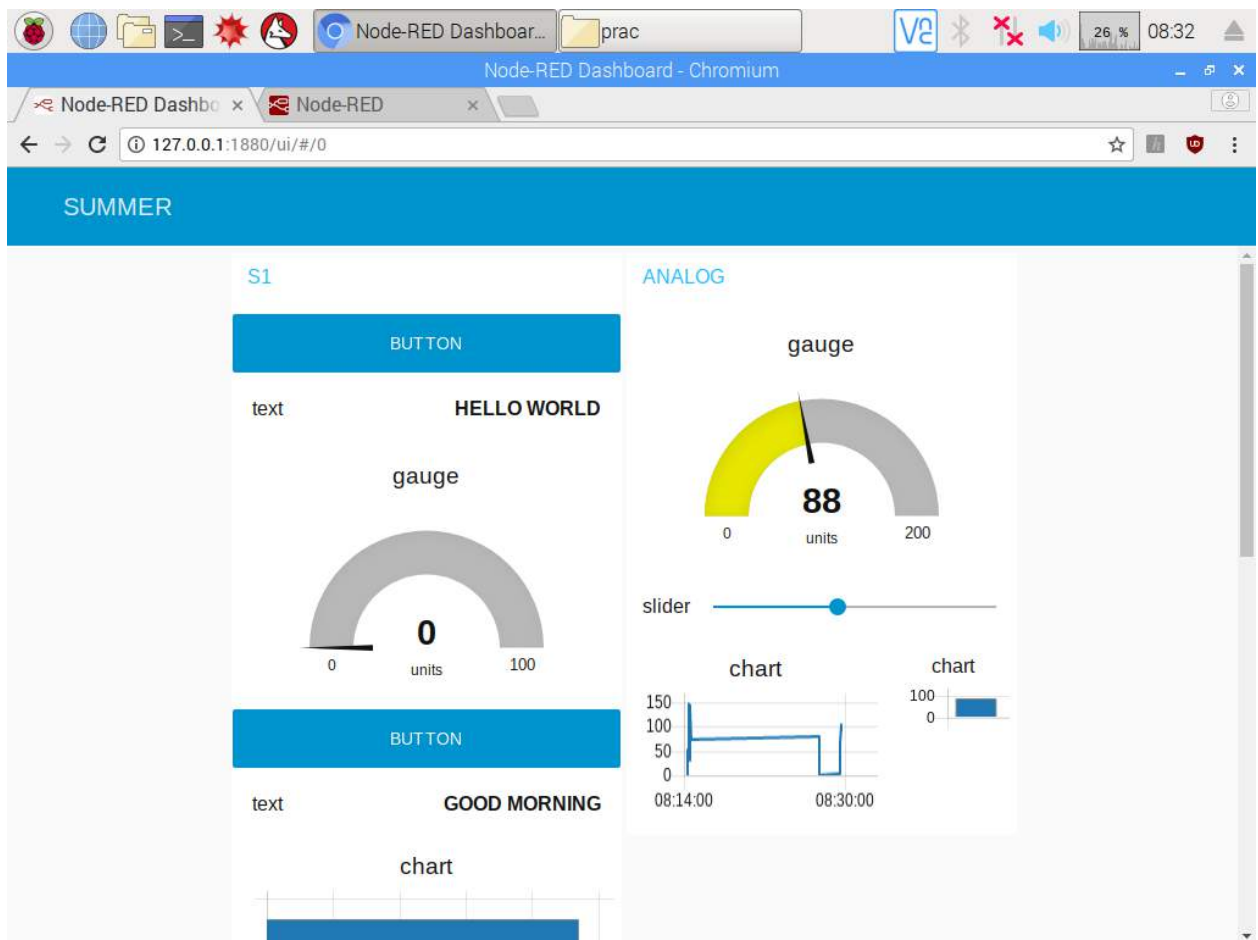
Use first colour for all bars

Series Colours

node settings







Practical 8:

MQTT

Aim : Set up the mosquitto MQTT server and client and write a python script to communicate data between Pi's.

Step 1 :

Update the system

Sudo apt-get update

Step 2 :

Update the system repositories

- a) Sudo wget <http://repo.mosquitto.org/debian/mosquitto-repo.gpg.key>
- b) Sudo apt-key add mosquitto-repo.gpg.key
- c) cd/etc/apt/sources.list.d/
- d) sudo wget <http://repo.mosquitto.org/debian/mosquitto-wheezy.list>
- e) sudo apt-get update
- f) sudo apt-get install mosquitto

step 3:

install three parts of mosquitto proper

☒ sudo apt-get install mosquitto mosquitto-client python-mosquitto

step 4:

stop the server

☒ sudo /etc/init.d/mosquitto stop

step 5 :

Configuring and starting with the mosquitto

- ☒ sudo nano/etc/mosquitto/mosquitto.conf
- ☒ log-dest topic
- ☒ log-type error
- ☒ log-type warning
- ☒ log-type notice

- ☒ log-type information
- ☒ connection_message true
- ☒ log-timestamp true
- ☒ include_dir/etc/mosquitto/conf.d

step 6 : Now start server

- ☒ sudo etc/init.d/mosquitto start

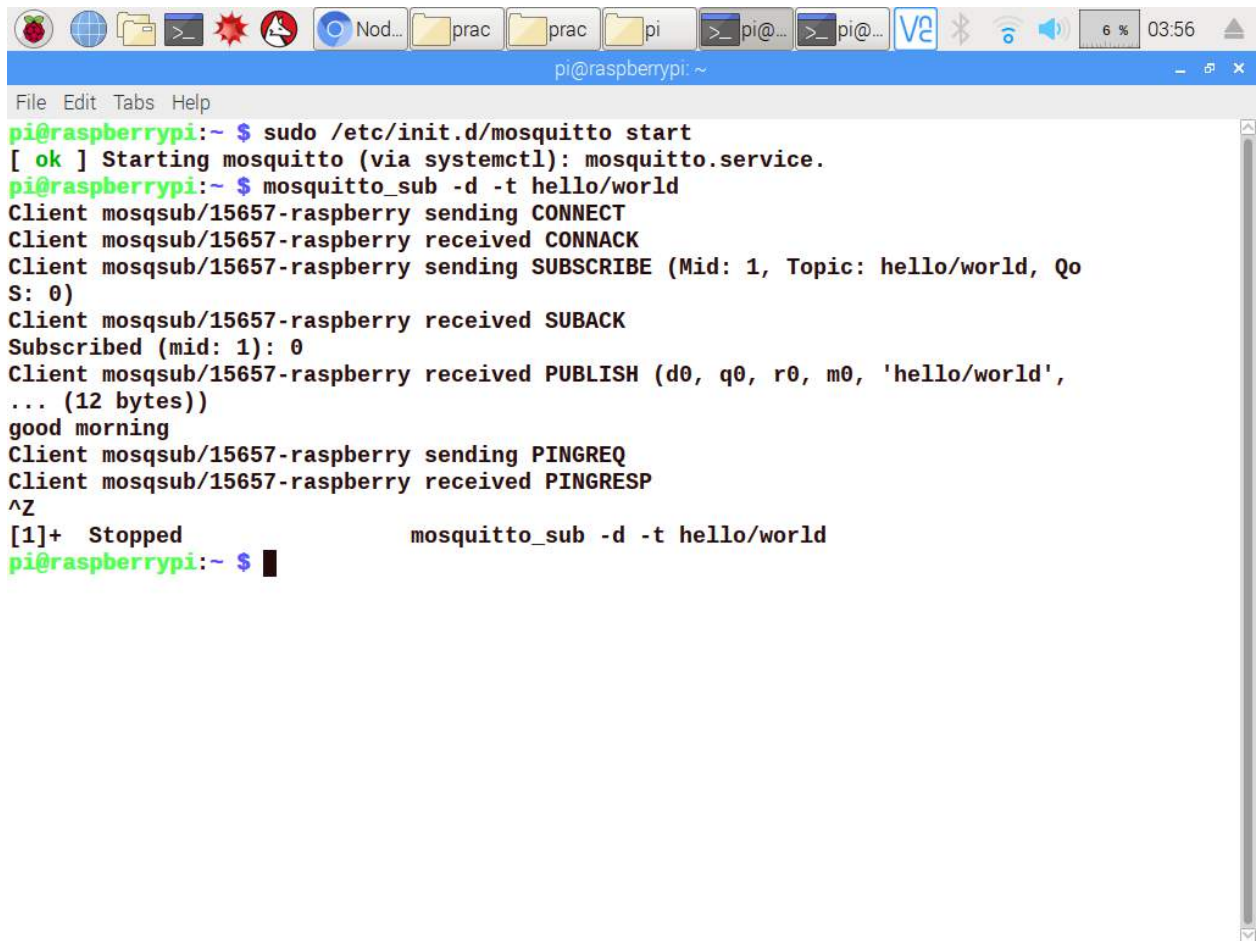
step 7 : Open two terminals using putty

a) At terminal 1 type the type following

- ☒ mosquitto_sub -d -t hello/world

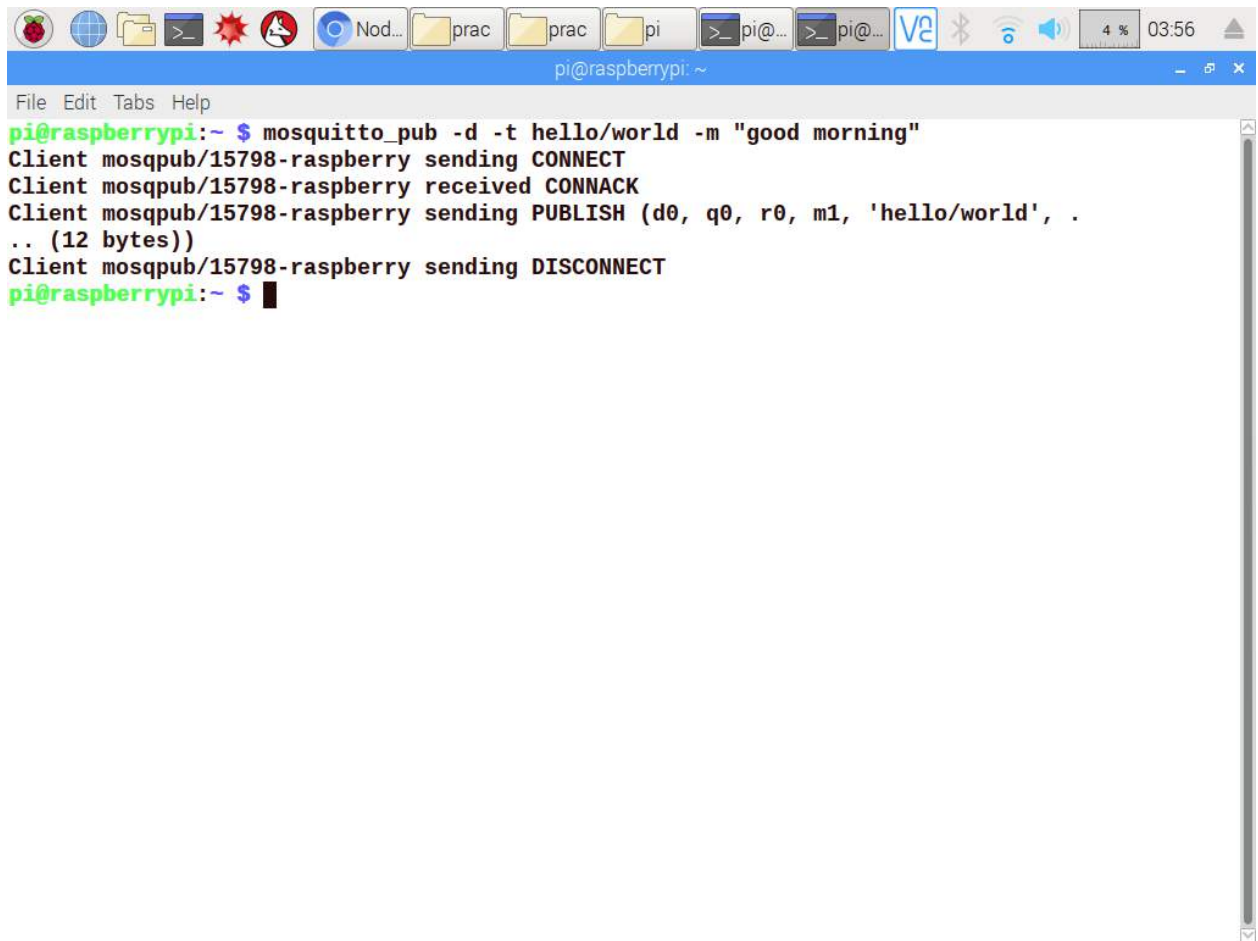
b) At terminal 2 type following

- ☒ Mosquitto_pub -d -t hello/world -m "good morning"



The image shows a terminal window on a Raspberry Pi. The top bar displays various system icons including a Raspberry Pi logo, network status, battery level at 6%, and the time 03:56. The terminal title bar reads 'pi@raspberrypi: ~'. The terminal content shows the following sequence of commands and outputs:

```
pi@raspberrypi:~ $ sudo /etc/init.d/mosquitto start
[ ok ] Starting mosquitto (via systemctl): mosquitto.service.
pi@raspberrypi:~ $ mosquitto_sub -d -t hello/world
Client mosqsub/15657-raspberry sending CONNECT
Client mosqsub/15657-raspberry received CONNACK
Client mosqsub/15657-raspberry sending SUBSCRIBE (Mid: 1, Topic: hello/world, QoS: 0)
Client mosqsub/15657-raspberry received SUBACK
Subscribed (mid: 1): 0
Client mosqsub/15657-raspberry received PUBLISH (d0, q0, r0, m0, 'hello/world', ... (12 bytes))
good morning
Client mosqsub/15657-raspberry sending PINGREQ
Client mosqsub/15657-raspberry received PINGRESP
^Z
[1]+  Stopped                  mosquitto_sub -d -t hello/world
pi@raspberrypi:~ $
```



The image shows a terminal window on a Raspberry Pi. The window has a title bar with the text "pi@raspberrypi: ~" and standard window controls. The terminal output shows the execution of the command "mosquitto_pub -d -t hello/world -m 'good morning'". This triggers a series of MQTT protocol messages: a CONNECT from the client, a CONNACK received by the client, a PUBLISH of the message "good morning" to the topic "hello/world", and finally a DISCONNECT from the client. The terminal prompt is "pi@raspberrypi:~ \$".

```
pi@raspberrypi:~ $ mosquitto_pub -d -t hello/world -m "good morning"
Client mosqpub/15798-raspberry sending CONNECT
Client mosqpub/15798-raspberry received CONNACK
Client mosqpub/15798-raspberry sending PUBLISH (d0, q0, r0, m1, 'hello/world', .
.. (12 bytes))
Client mosqpub/15798-raspberry sending DISCONNECT
pi@raspberrypi:~ $
```