**Practical No-1**

**1a) Edit text files with nano and cat editor, Learn sudo privileges and Unix shell commands such as cd , ls , cat, etc**

* + 1. **sudo command**
* Short for superuser do**,** **sudo**is one of the most popular basic Linux commands that lets you perform tasks that require administrative or root permissions.
* Command:

**sudo bash**

**sudo apt install update/upgrade**

**sudo raspi-config**

**sudo apt install (any-package-name)**

**sudo reboot**

### ls command

* The**ls** command lists files and directories within a system. Running it without a flag or parameter will show the current working directory’s content.
* Command:

**ls**

### mkdir command

### Use the ****mkdir**** command to create one or multiple directories at once and set permissions for each of them. The user executing this command must have the privilege to make a new folder in the parent directory, or they may receive a permission denied error.

* Command:

**mkdir directory\_name**

### cat command

* Concatenate, or **cat**, is one of the most frequently used Linux commands. It lists, combines, and writes file content to the standard output. To run the cat command, type **cat** followed by the file name and its extension.
* Command:

**cat > filename.txt**

### cd command

* To navigate through the Linux files and directories, use the **cd** command. Depending on your current working directory, it requires either the full path or the directory name.
* Command:

**cd (folder\_name)**

**cd ..**

* + 1. [**mv command**](https://www.hostinger.com/tutorials/how-to-rename-files-in-linux/)
* The primary use of the [**mv command**](https://www.hostinger.com/tutorials/how-to-rename-files-in-linux/) is to move and rename files and directories. Additionally, it doesn’t produce an output upon execution.
* Command:

**mv old\_filename.txt new\_filename.txt**

### rm command

* The **rm** command is used to delete files within a directory. Make sure that the user performing this command has write permissions.
* Command:

**rm filename.txt**

### grep command

* Another basic Linux command on the list is **grep**or global regular expression print. It lets you find a word by searching through all the texts in a specific file.
* Once the [**grep command**](https://www.hostinger.com/tutorials/grep-command-in-linux-useful-examples/) finds a match, it prints all lines that contain the specific pattern. This command helps filter through large log files.
* For example, you want to search for the word **blue**in the **notepad.txt**
* Command:

**grep blue notepad.txt**

### ping command

* **The**[**ping command**](https://www.hostinger.com/tutorials/ping-an-ip) is one of the most used basic Linux commands for checking whether a network or a server is reachable. In addition, it is used to troubleshoot various connectivity issues.
* Command:

**ping [option] [hostname\_or\_IP\_address]**

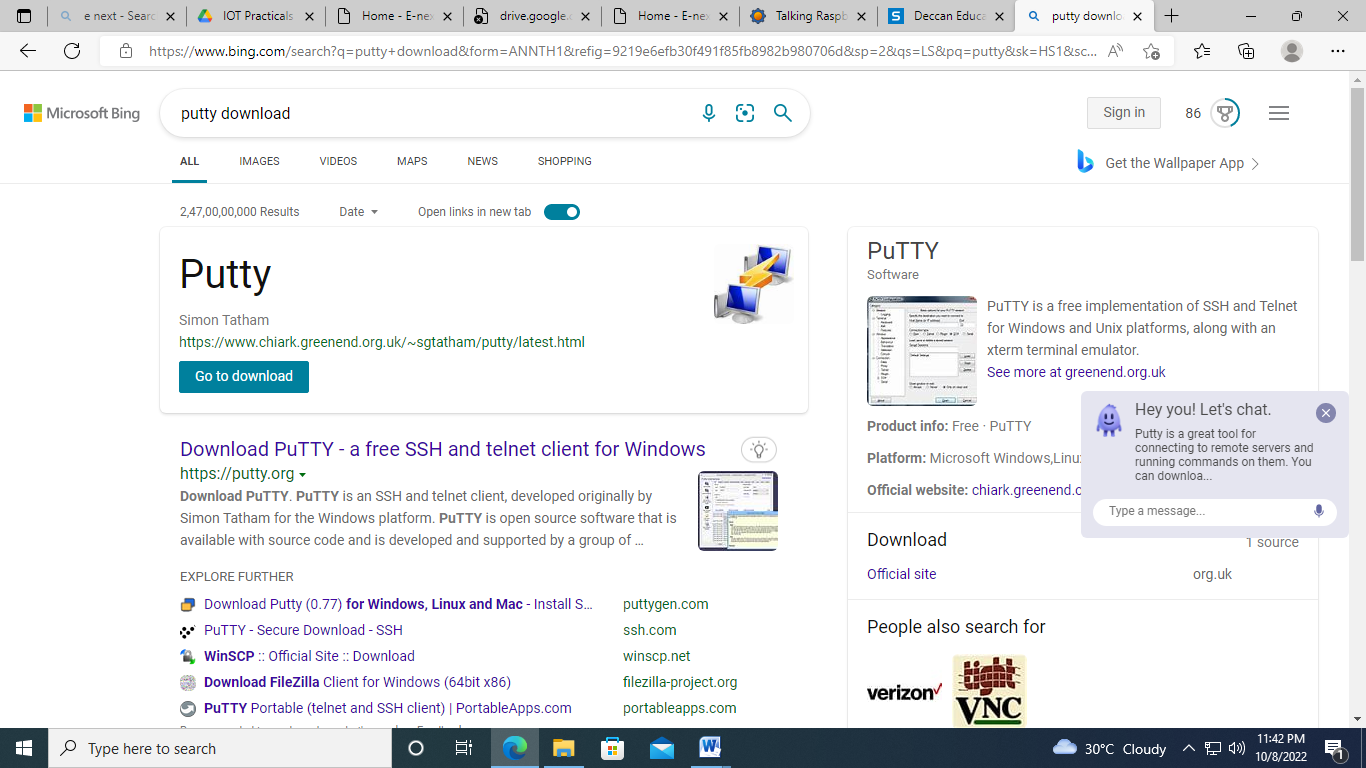
### nano

* The [**nano command**](https://www.hostinger.com/tutorials/how-to-install-and-use-nano-text-editor) denotes keywords and can work with most languages.
* Command:

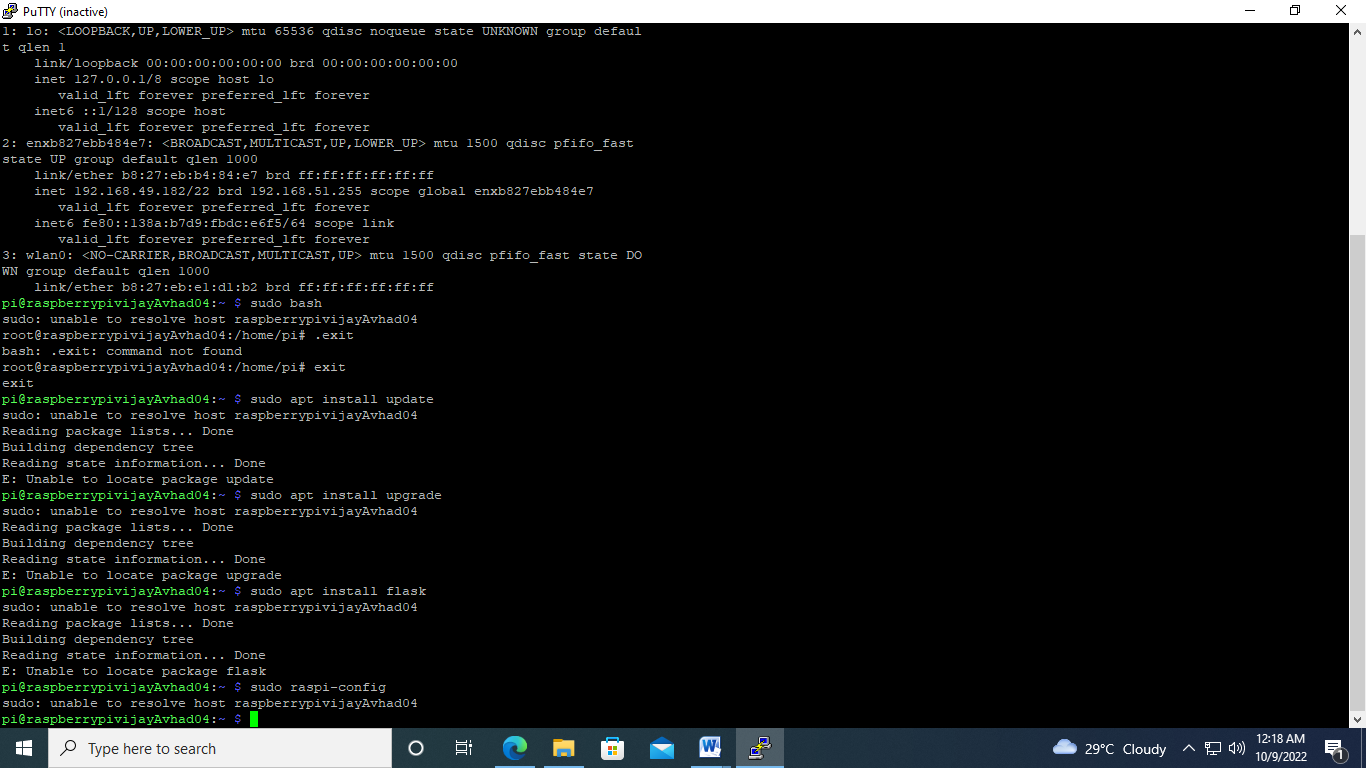
**nano filename.txt/.py**

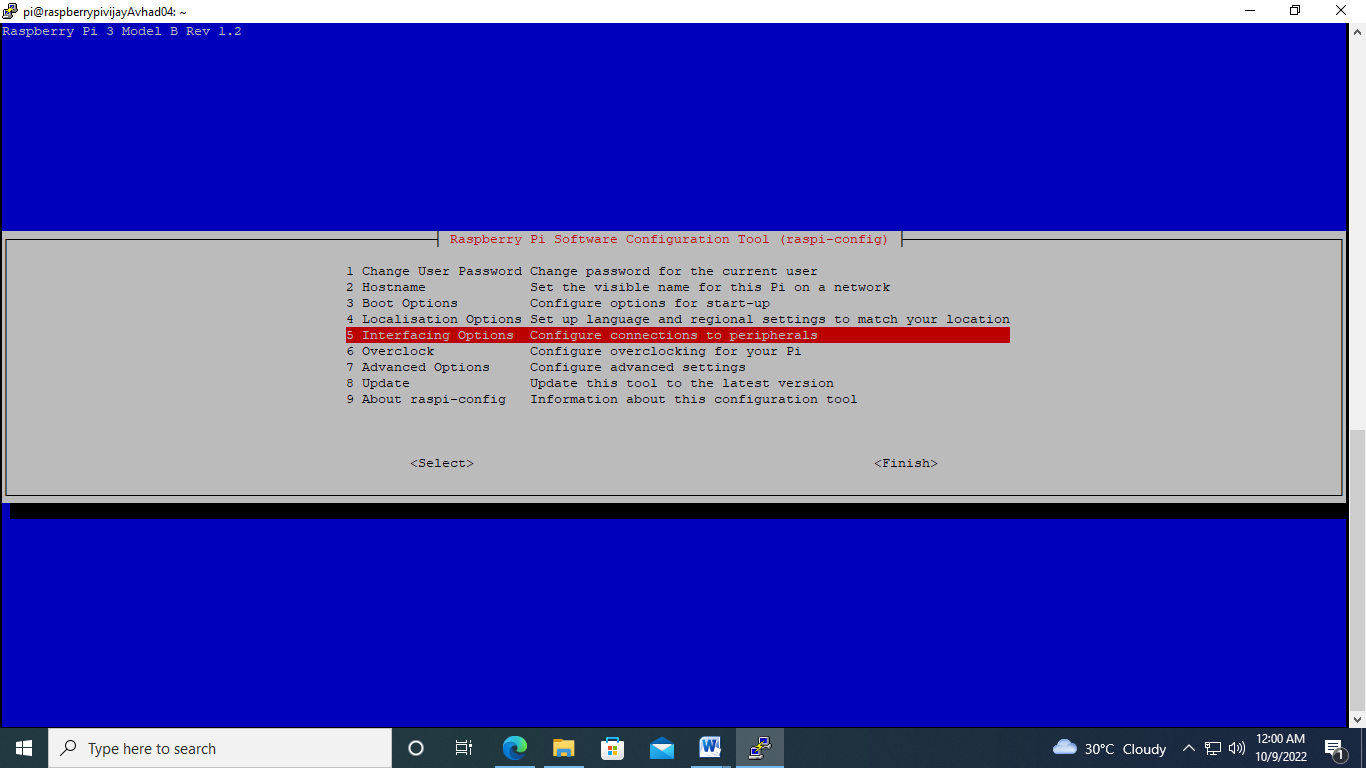
**1b) Learn to set dynamic and static IP. Connect to and Ethernet and WiFi network. Learn to vnc and ssh into a raspberry pi using vnc and putty from a different computer on the network.**

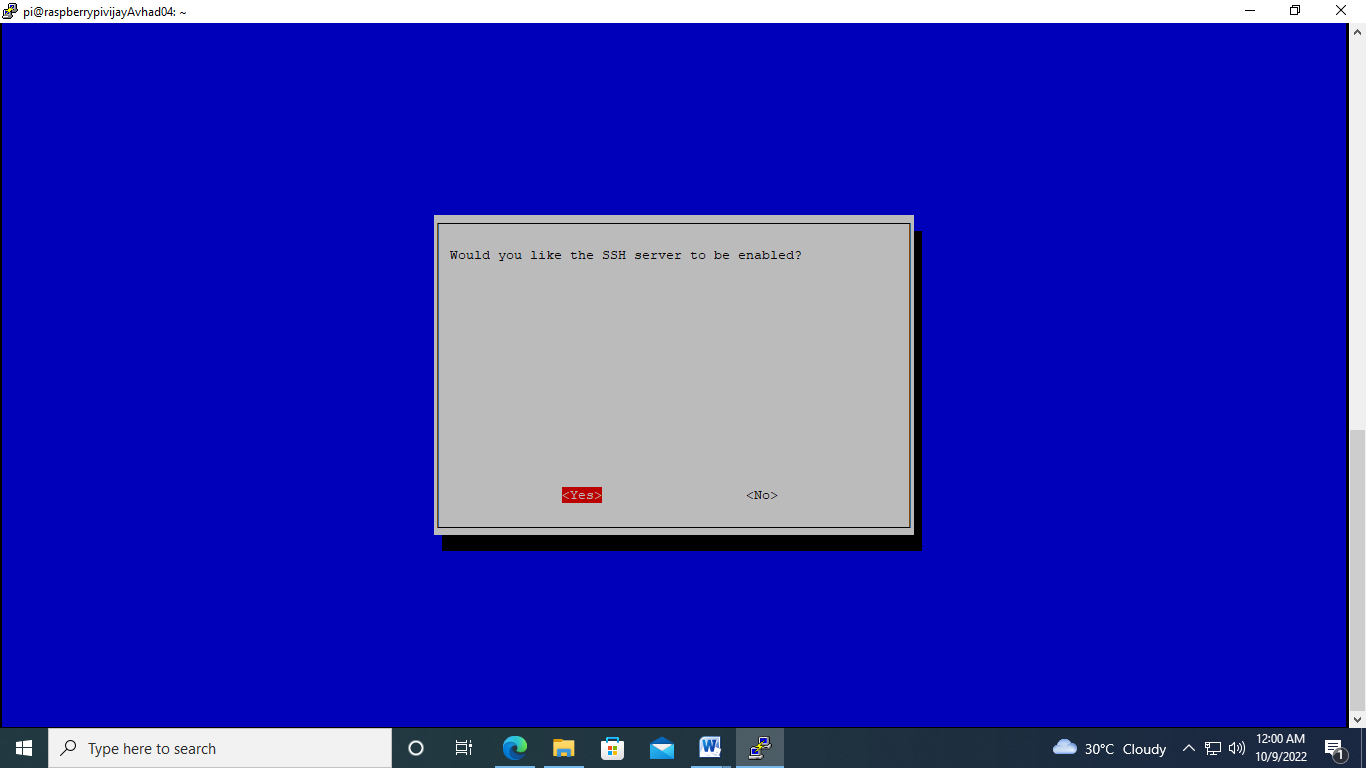
* For remote access of Raspberry pi from windows
* First Download putty on windows desktop

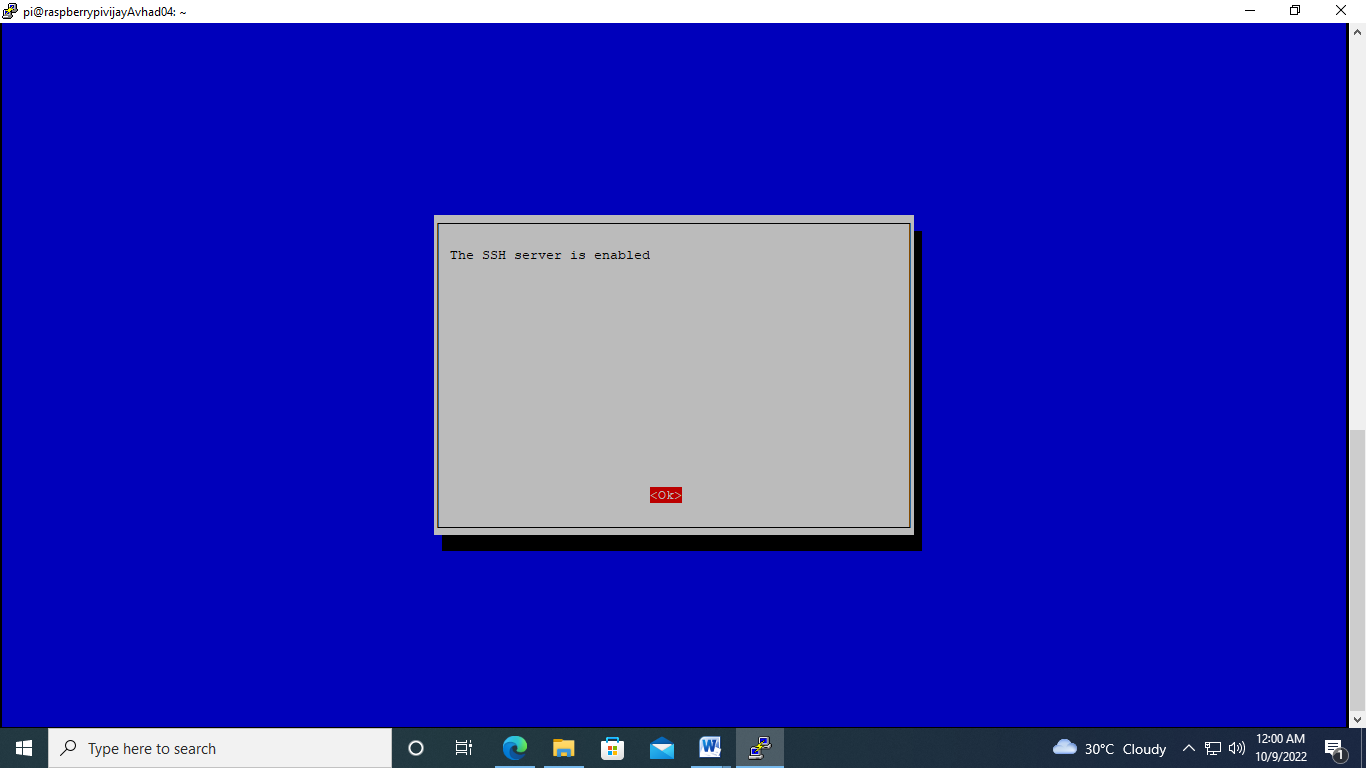


* Enable SSH in Raspberry pi with the help of **sudo raspi-config command.**
* After select interface option that would be the 5th option, click on enter
* In that click on SSH option, click for enabling it and then click on yes and then click on ok.
* Then click on Cancel



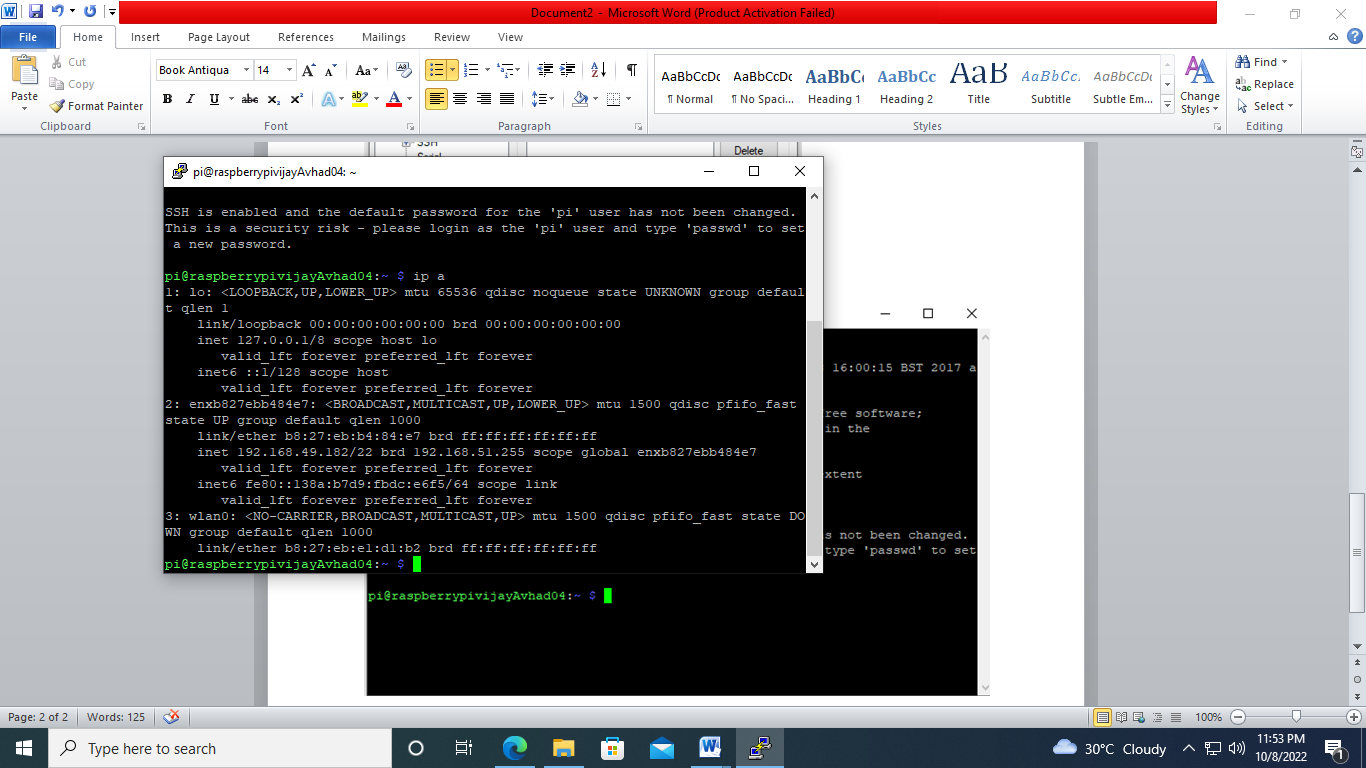


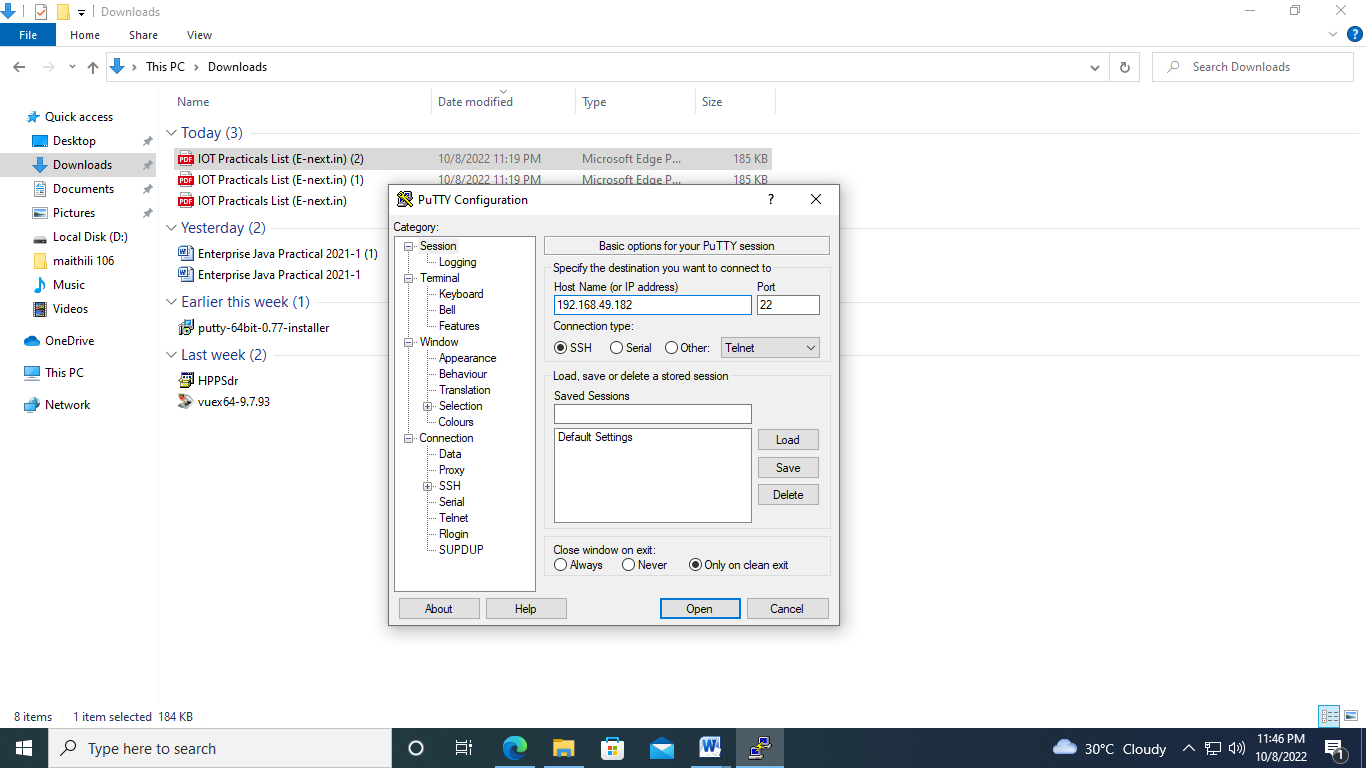




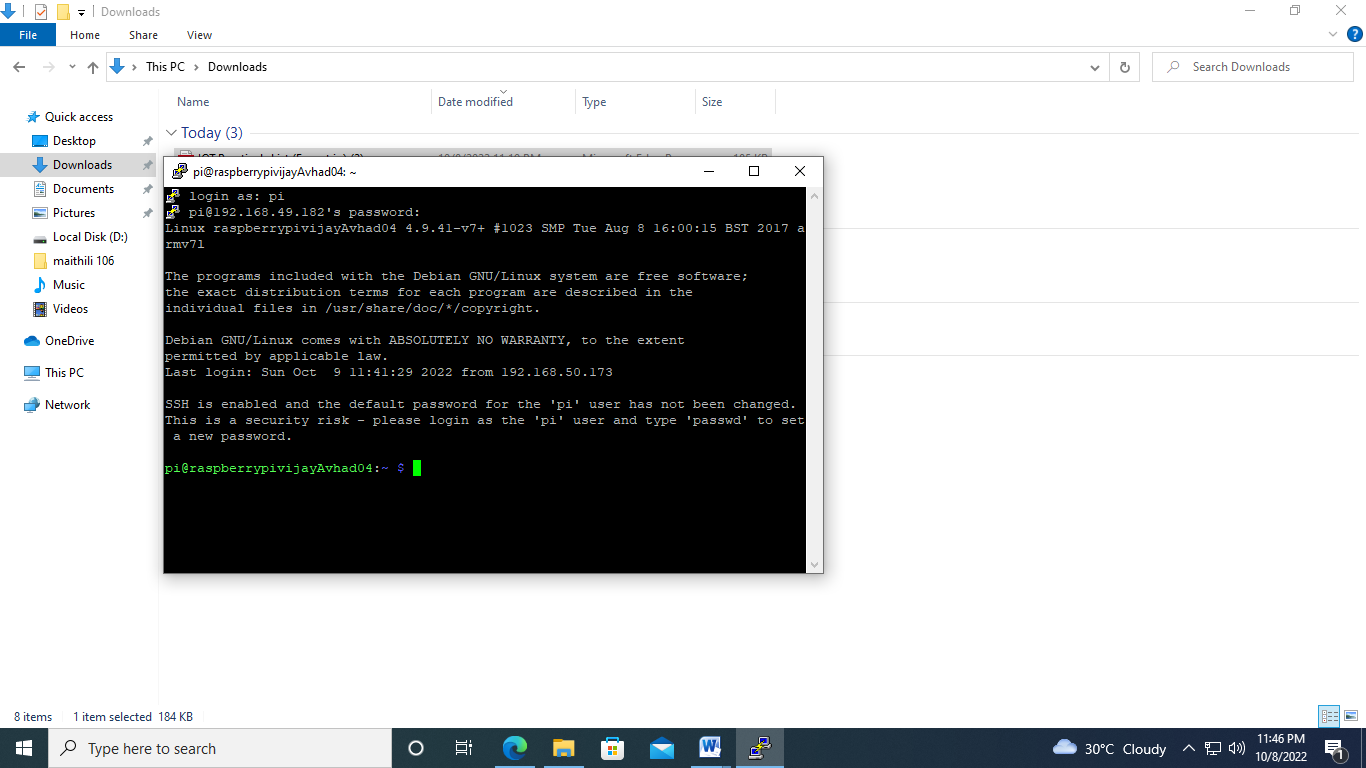
* Install putty on your windows, after that open putty and type the ip address of the raspberry pi in the host name (eg. 192.168.49.123)
* You can get ip address of your Raspberry pi with the help of two different commands in the Raspberry pi terminal

1. ifconfig
2. ip a





* Click on Accept
* Login as : **pi** and the password : **raspberry** (you won’t able to see the password but it is being typed)

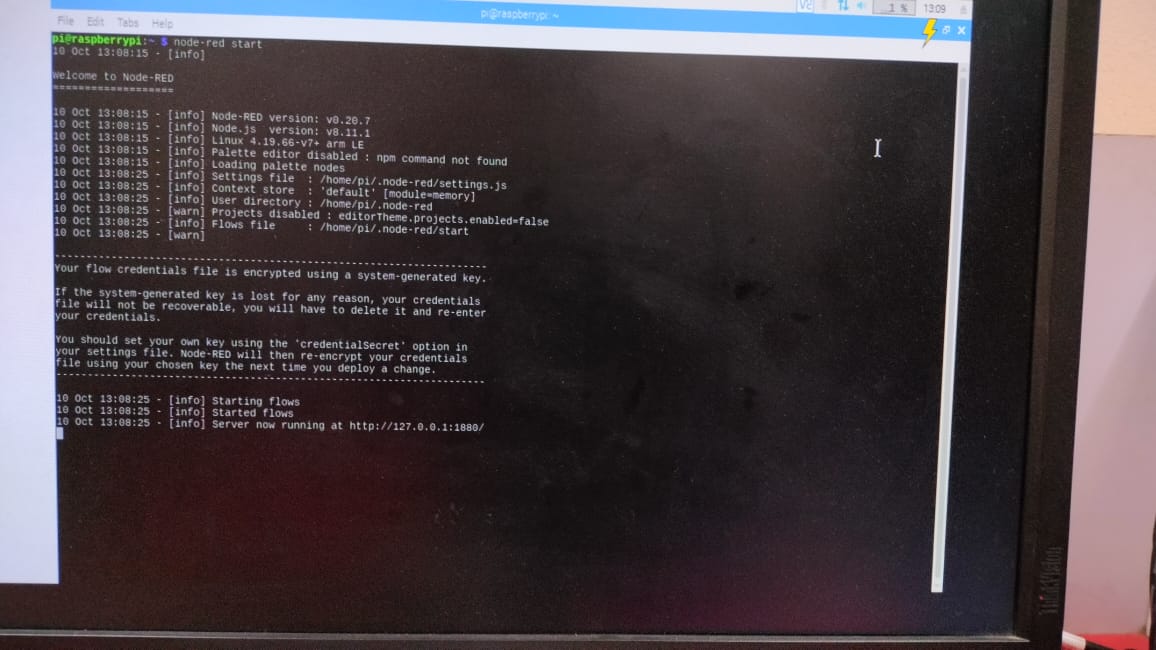


* The remote access has been successfully done.

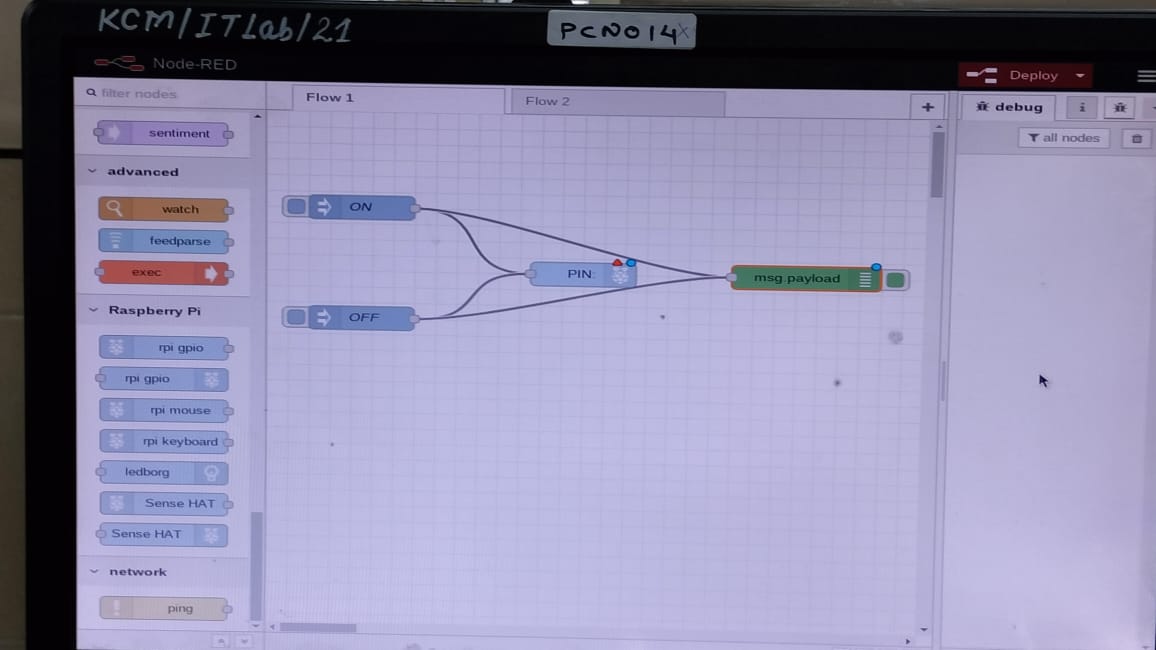
**PRACTICAL NO-2**

**Run the node red editor and run simple programs and trigger gpios. Use basic nodes such as inject, debug, gpios.**

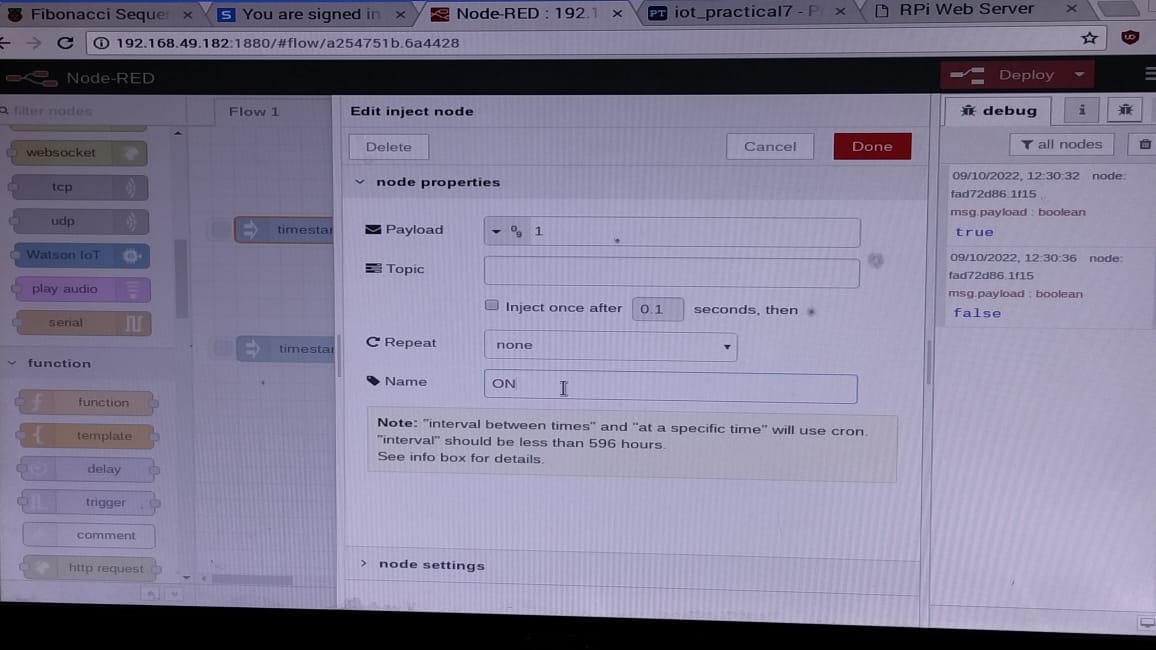
* Type **node-red start** command in the Raspberry Pi terminal to start the node-red

****

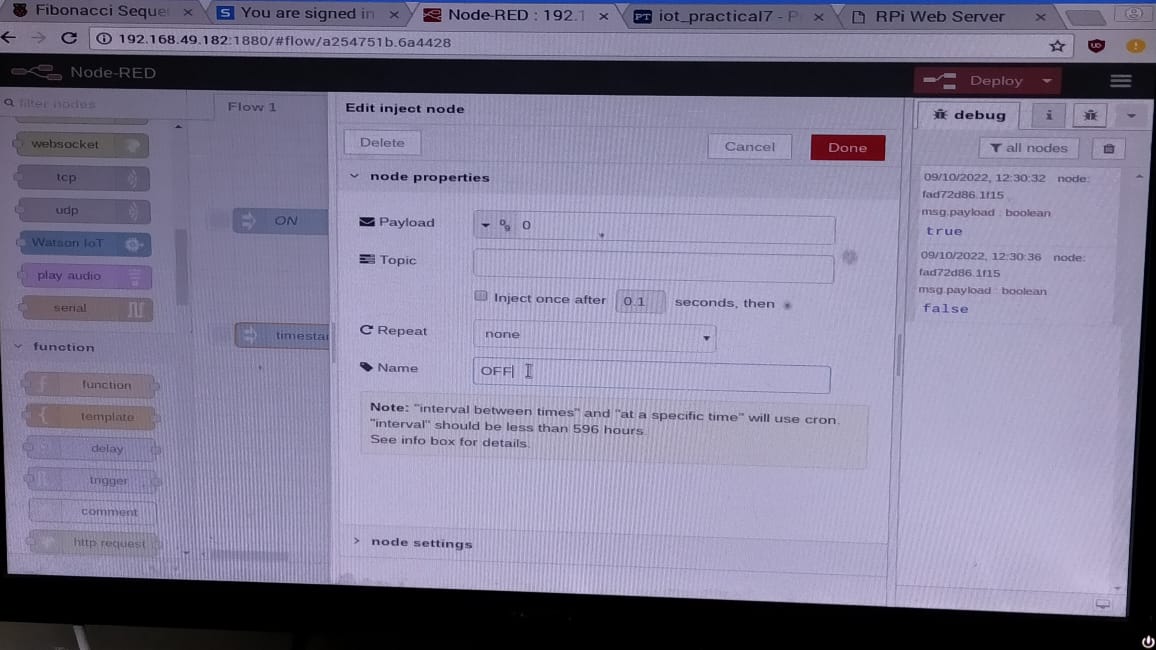
* Drag and drop **two inject** **node** from the input section, **one debug** from the output section **and one rpi gpio (2nd option)** from the raspberry pi section from the left-side panel.
* Make a connection accordingly as per the given diagram below



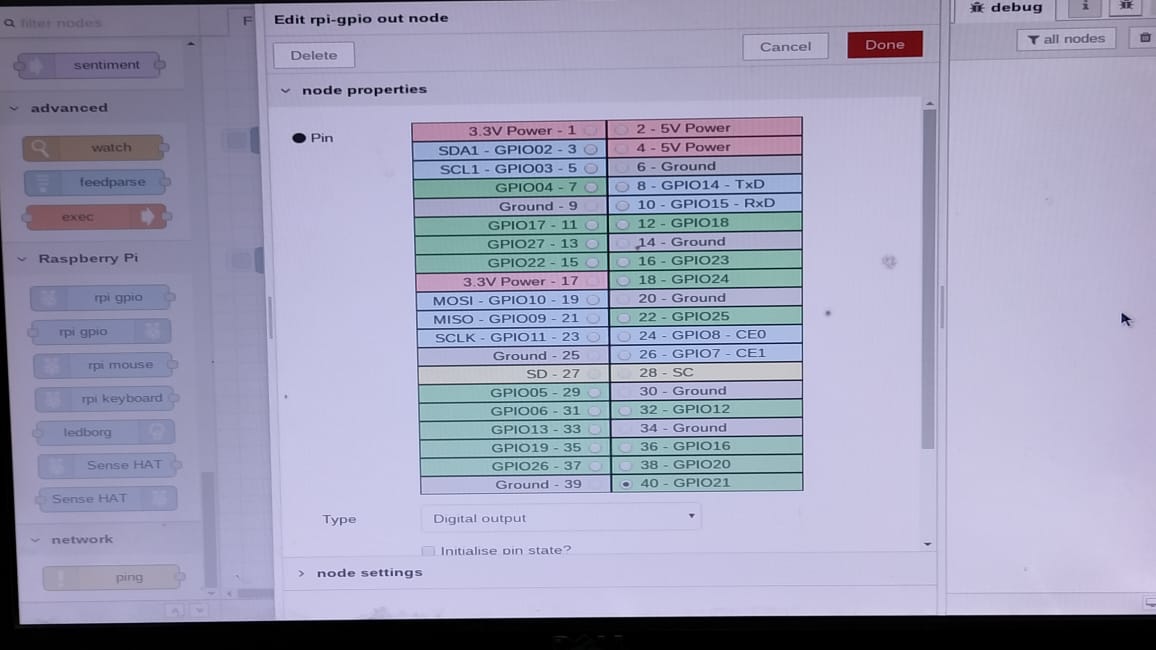
* Double click on the 1st inject node, it will display the properties of that inject node.
* In payload, click on the dropdown and select number and type 1 in the beside input and in name section type ON. Then click on Done.



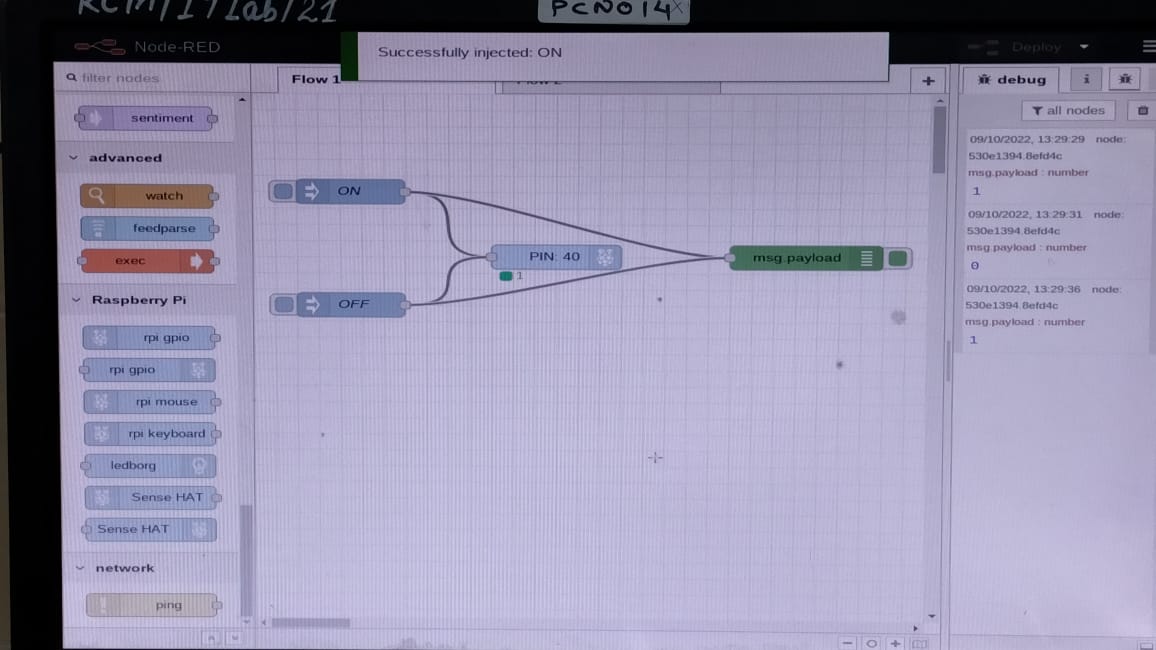
* Follow the same instructions for the 2nd inject node in payload select number, give input 0 and in name section type OFF. Then click on Done.

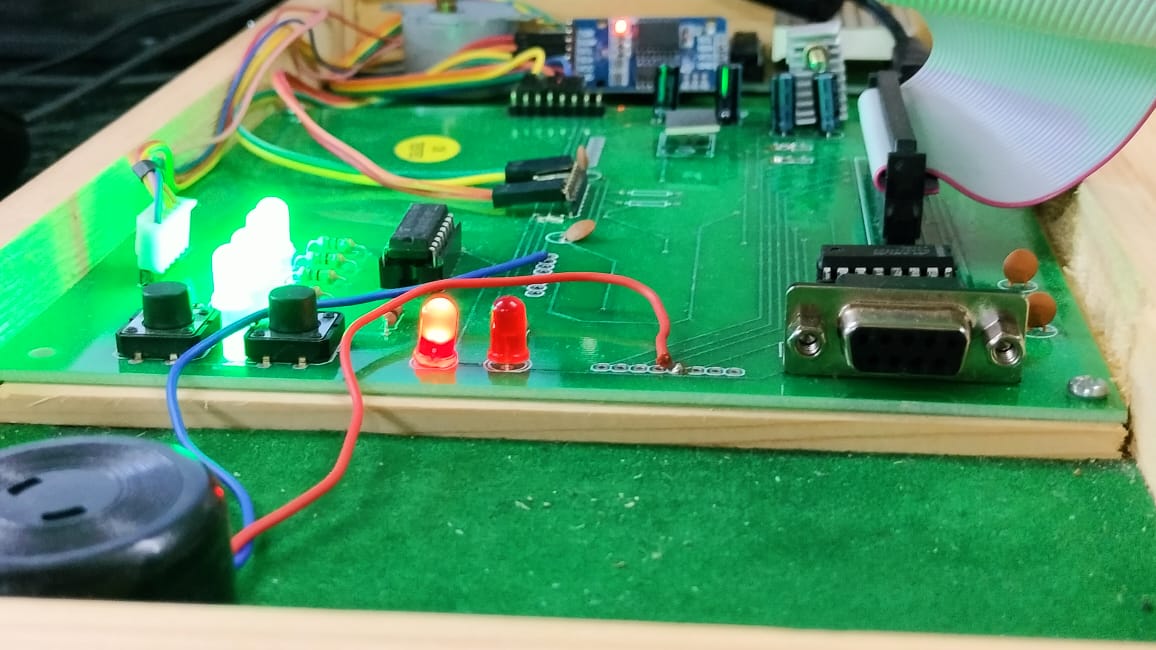


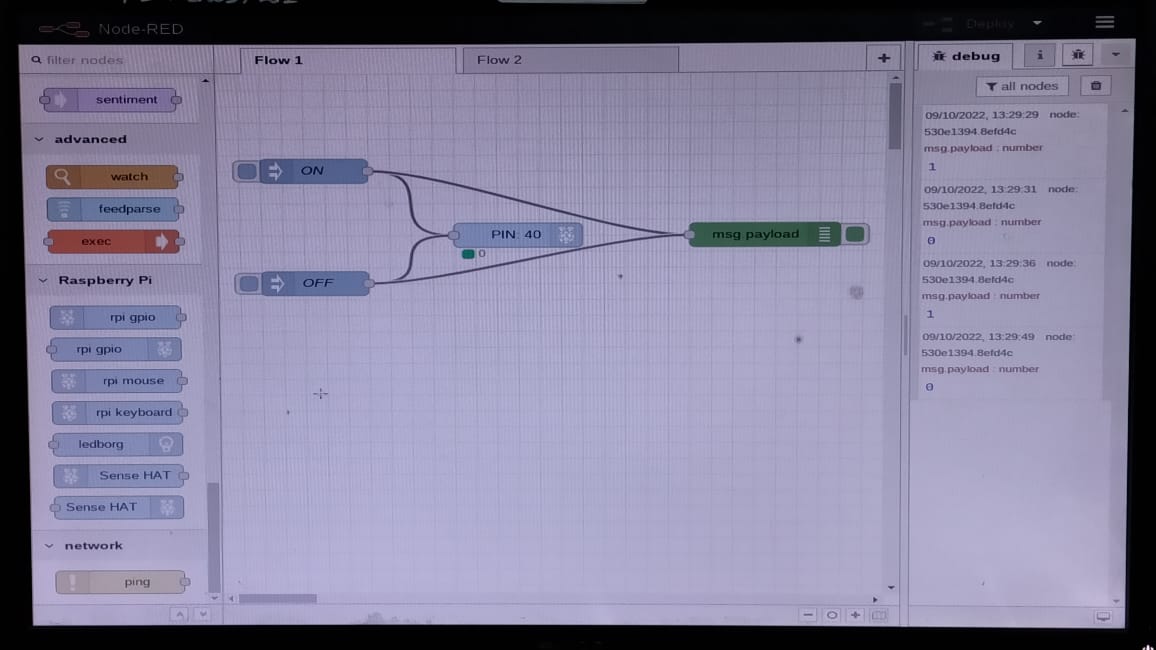
* Double click on the pin and select pin 40 or pin 38 as both pins are connection to led. Then click on Done.

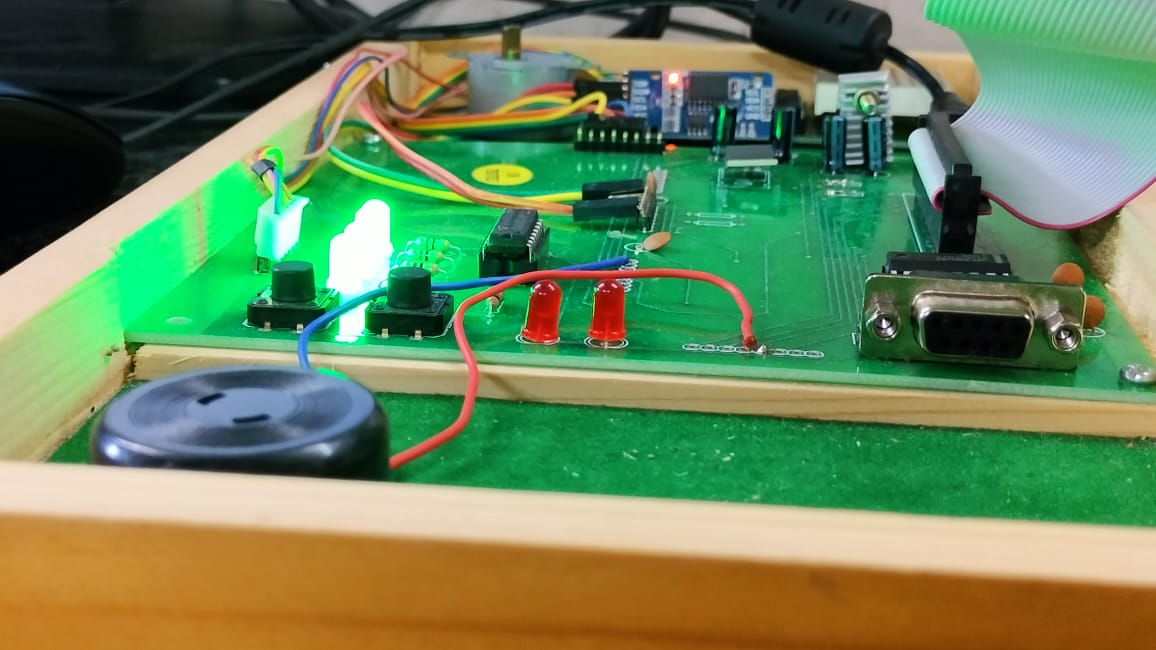


* Click on Deploy and after that click on the button beside ON and OFF node to see if the connection has been established successfully.
* You will be able to see the node outcome in debug pane on the right side of the screen







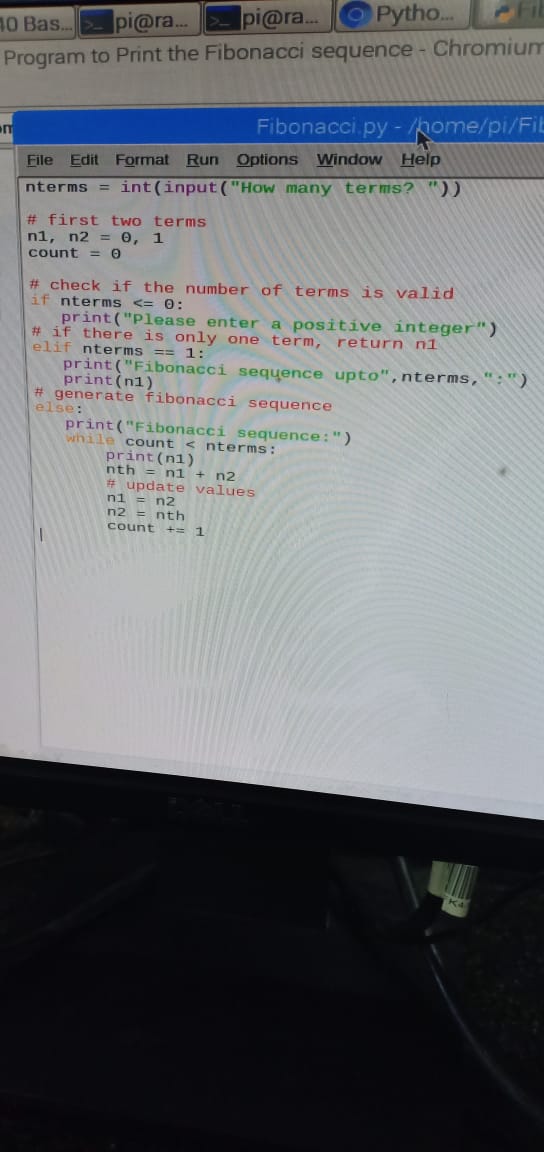


**PRACTICAL NO-3**

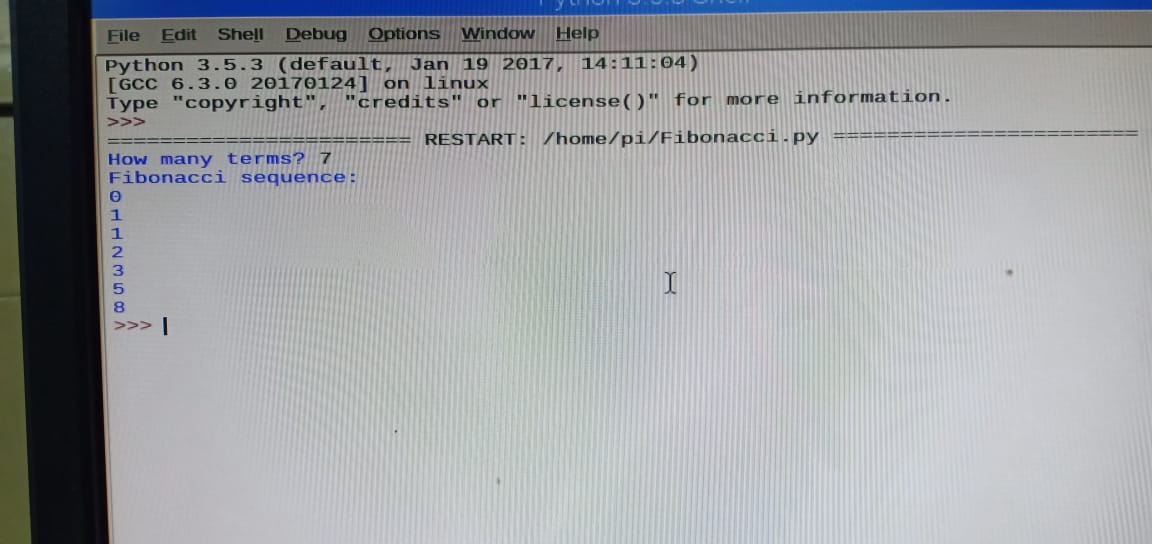
**Open the python idle editor and run simple Python scripts such as to print Fibonacci numbers, string functions. Learn how to install modules using Pip and write functions.**

* Click on the Raspberry pi icon.
* Click on Programming and Select Python IDLE.
* Open Python IDLE 3 or 2 in Raspberry pi, click on file and then click on new file and type the given code.
* Run the given code to get the output.

**CODE-**



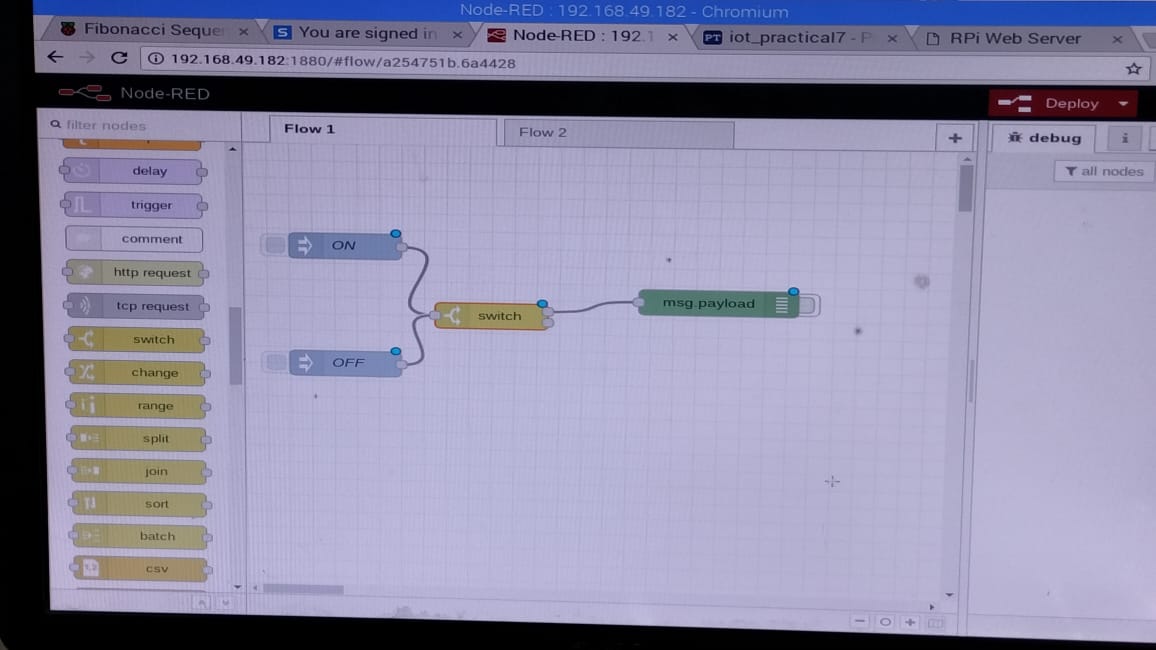
**OUTPUT-**



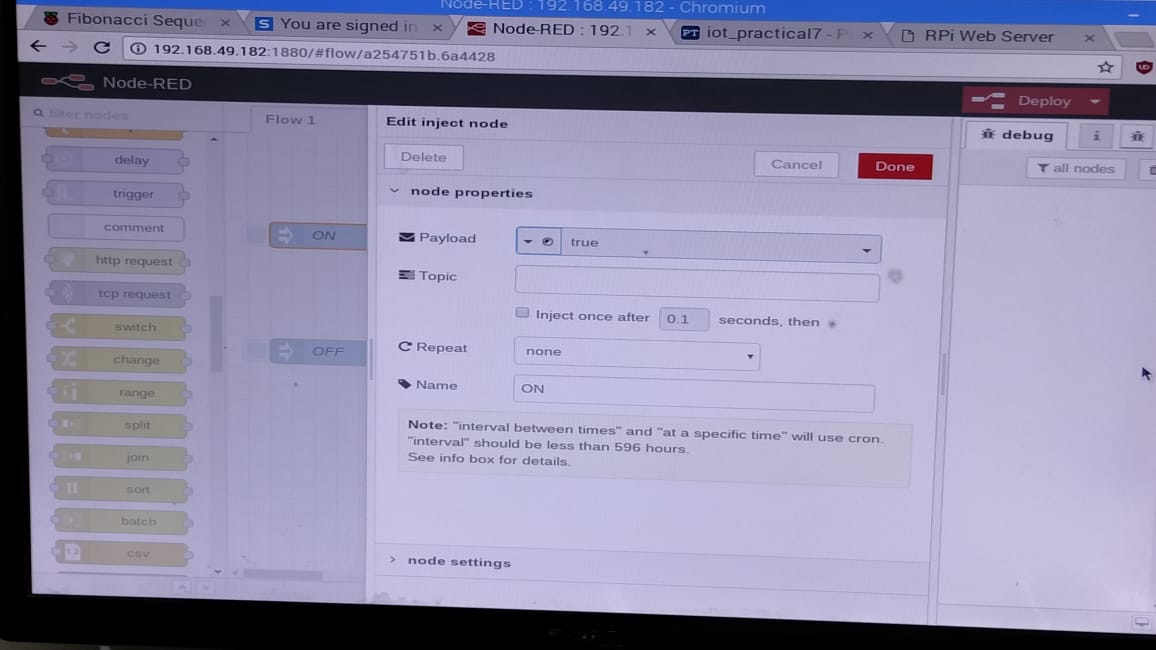
**PRACTICAL NO-4**

**Setup a physical button, switch and trigger a led in node red and python w debounce.**

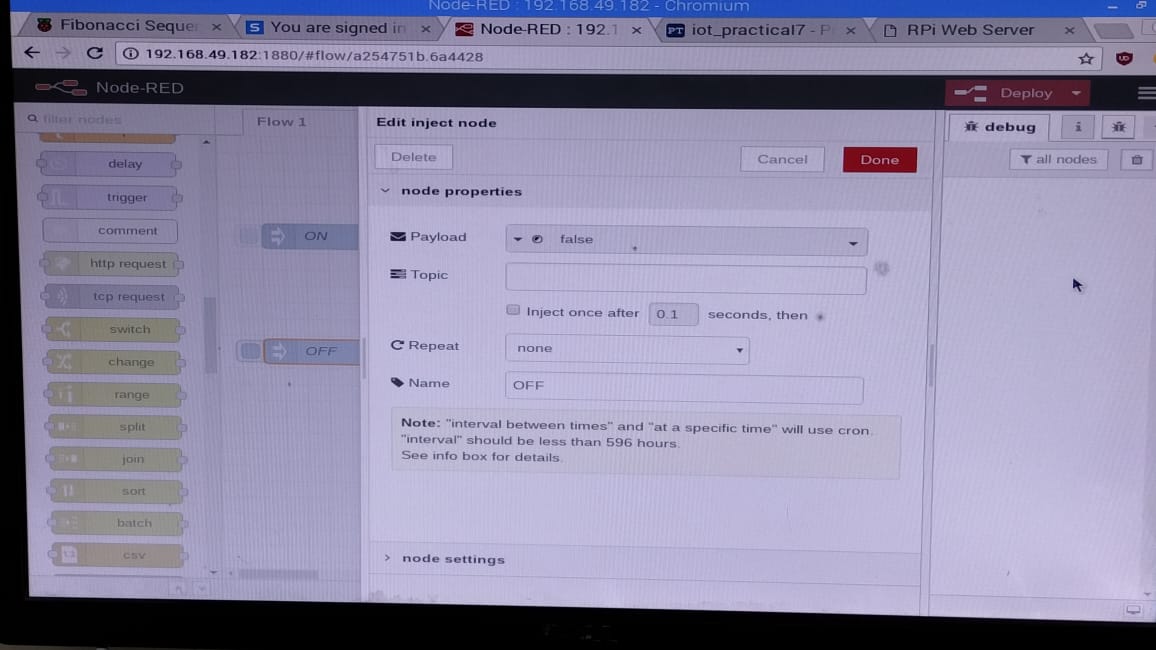
* **Switch**
* Drag and drop **two inject** **node** from the input section, **one debug** from the output section **and one switch** fromthe left side panel.
* Make a connection accordingly as per the given diagram below.



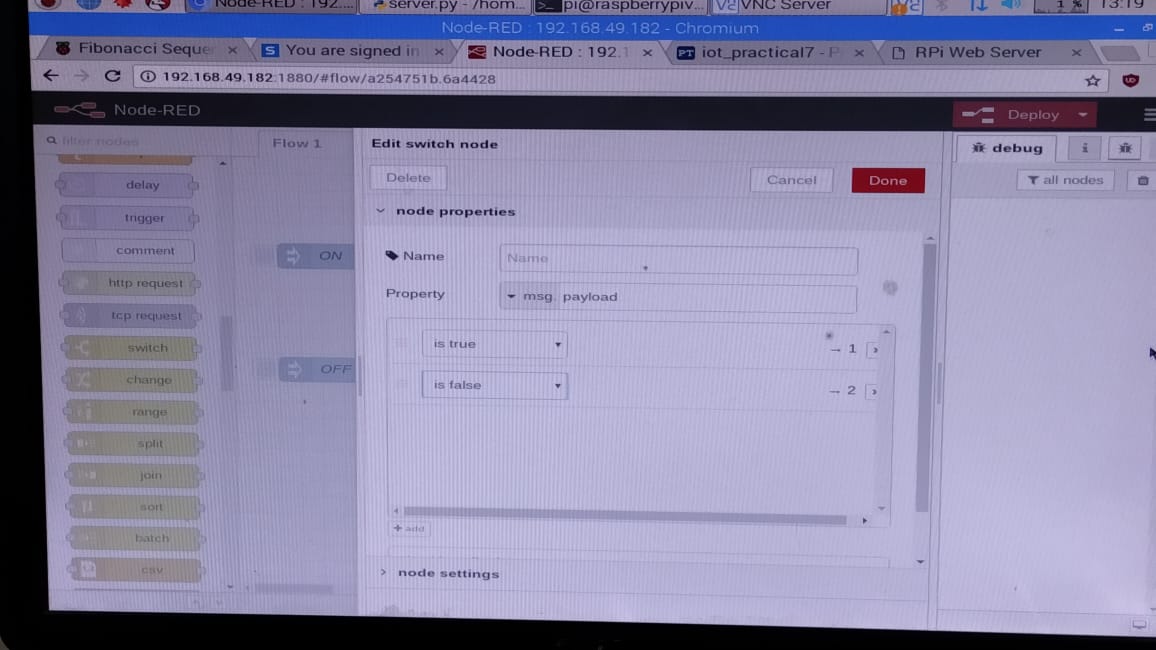
* Double click on the 1st inject node, it will display the properties of that inject node.
* In payload, click on the dropdown and select **Boolean** and **select true** in the beside input and in name section type ON. Then click on Done.



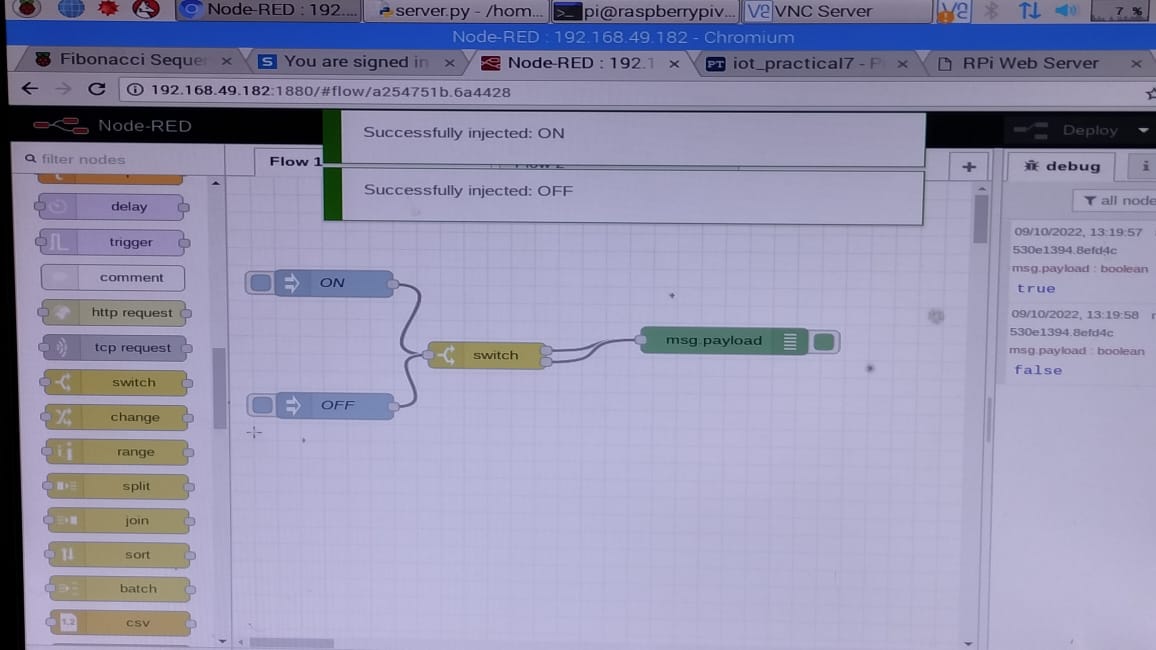
* Follow the same instructions for the 2nd inject node in payload select Boolean and select false in the beside input.
* In name section type OFF. Then click on Done.



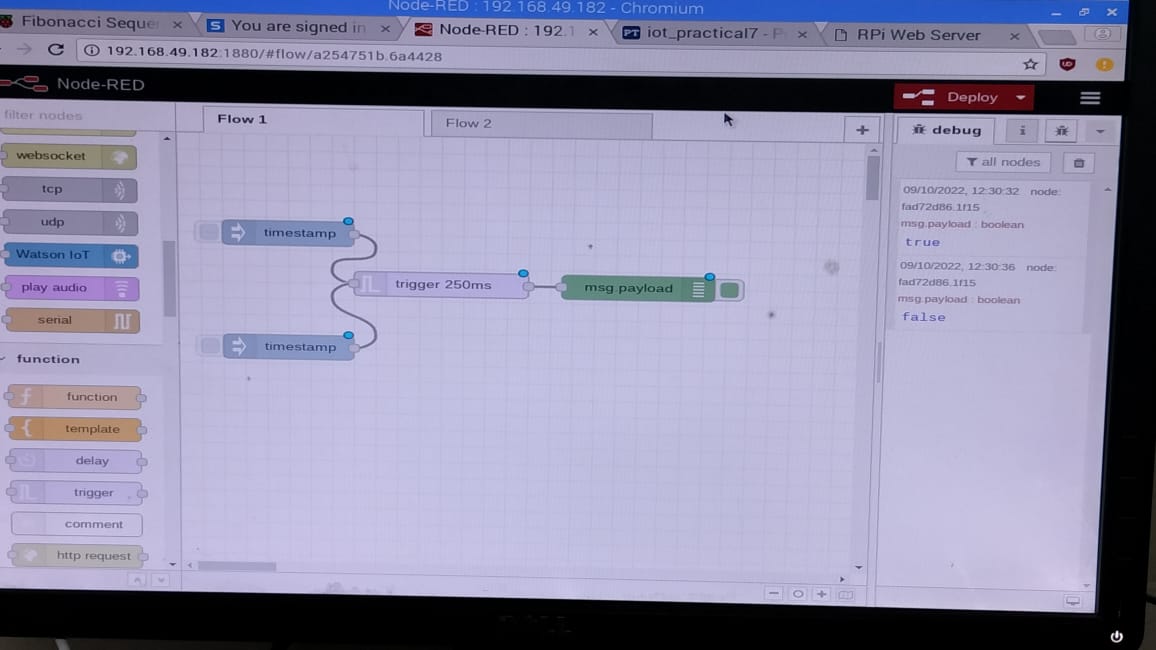
* Double click on switch node, in property pane instead of == click on the dropdown and select **is true** and then click on add button present on the left corner to add **is false** property.
* Click on Done.



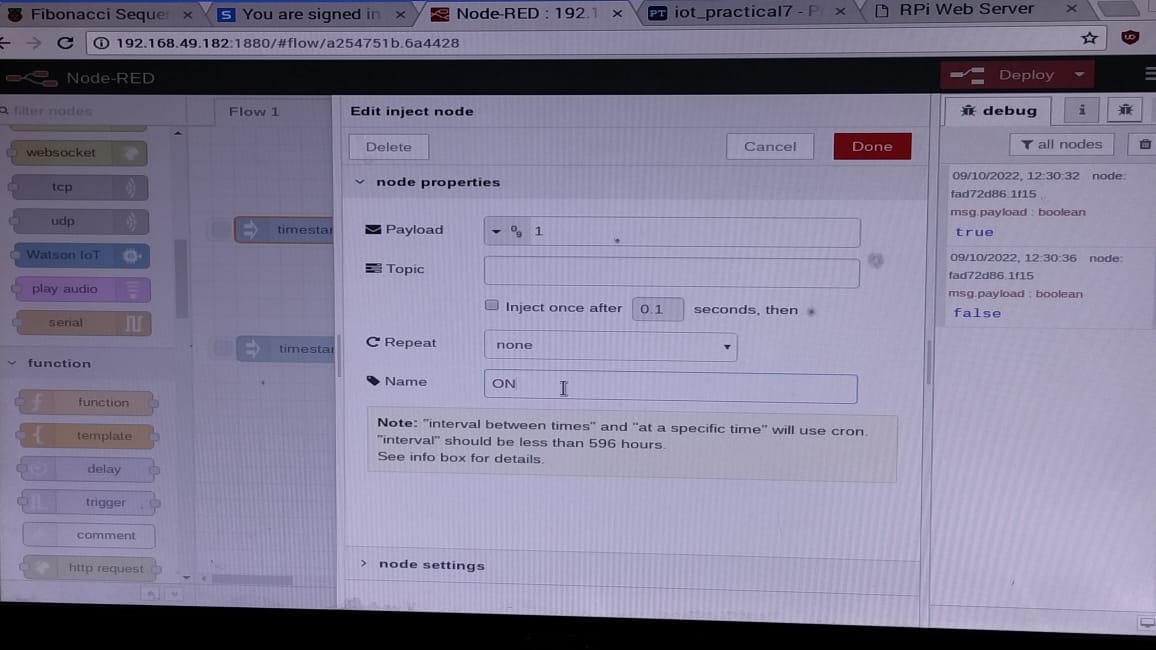
* Click on Deploy.
* After that click on the ON and OFF button which is present beside the respective node and the output is displayed in the debug section.



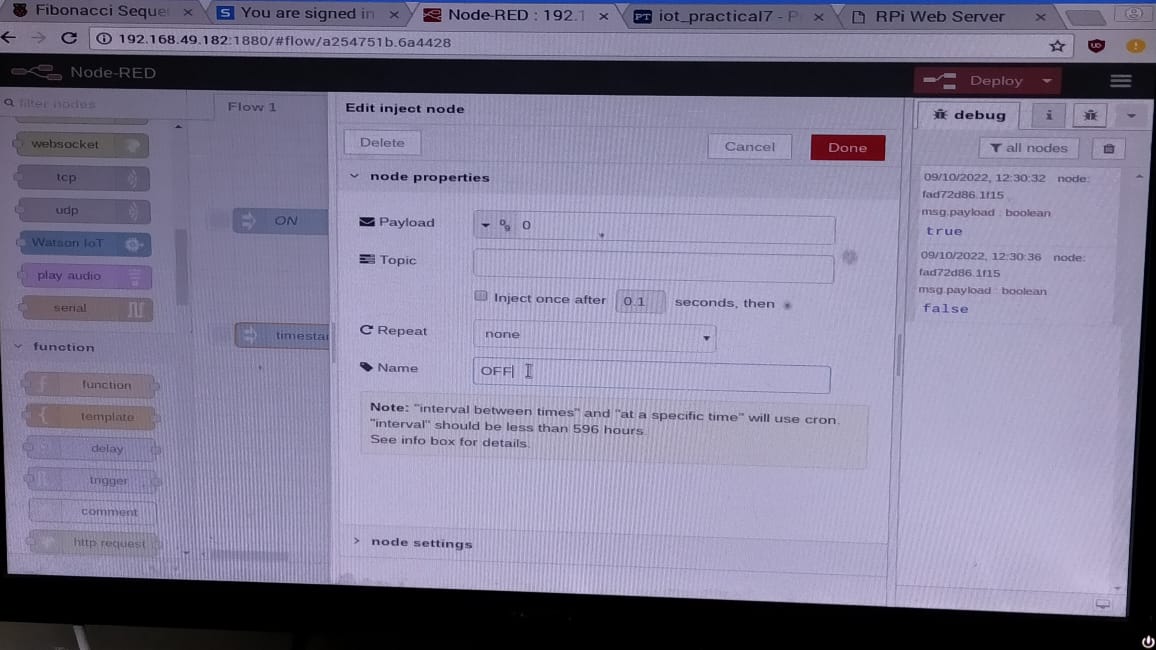
* **Trigger**
* Drag and drop **two inject** **node** from the input section, **one debug** from the output section **and one trigger** fromthe left side panel.
* Make a connection accordingly as per the given diagram below.



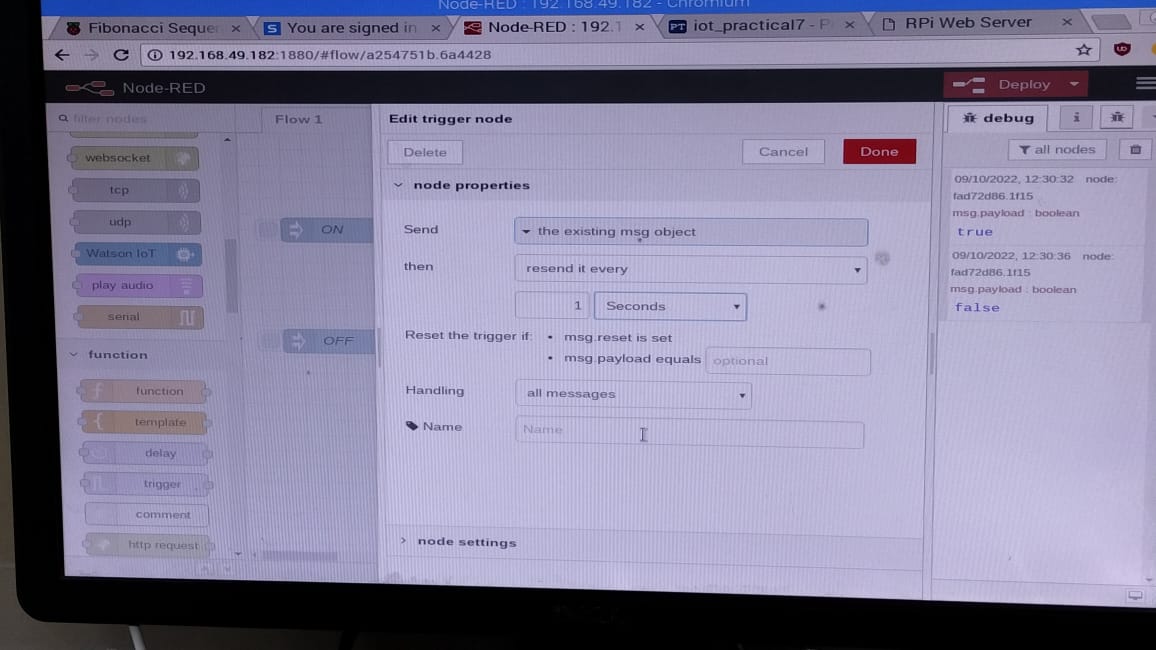
* Double click on the 1st inject node, it will display the properties of that inject node.
* In payload, click on the dropdown and select number and type 1 in the beside input and in name section type ON. Then click on Done.



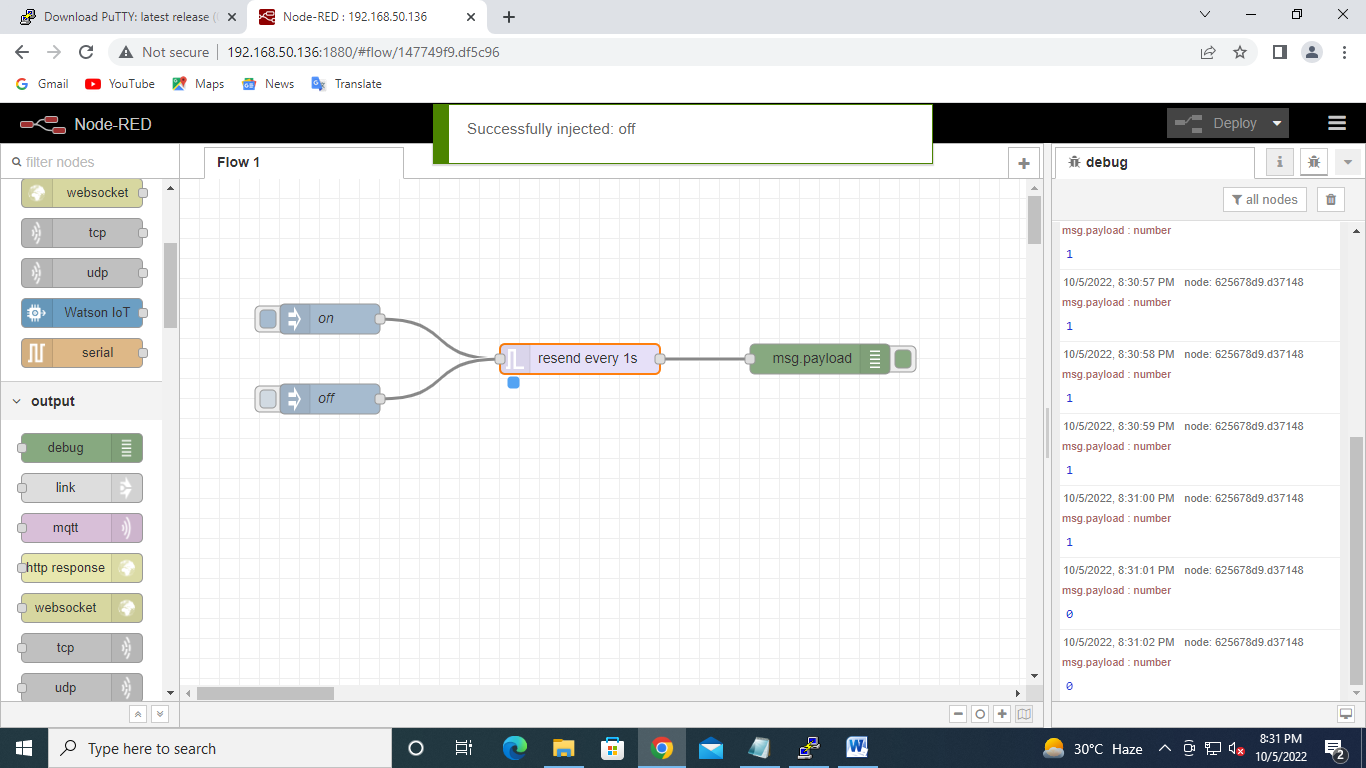
* Follow the same instructions for the 2nd inject node in payload select number, give input 0 and in name section type OFF. Then click on Done.

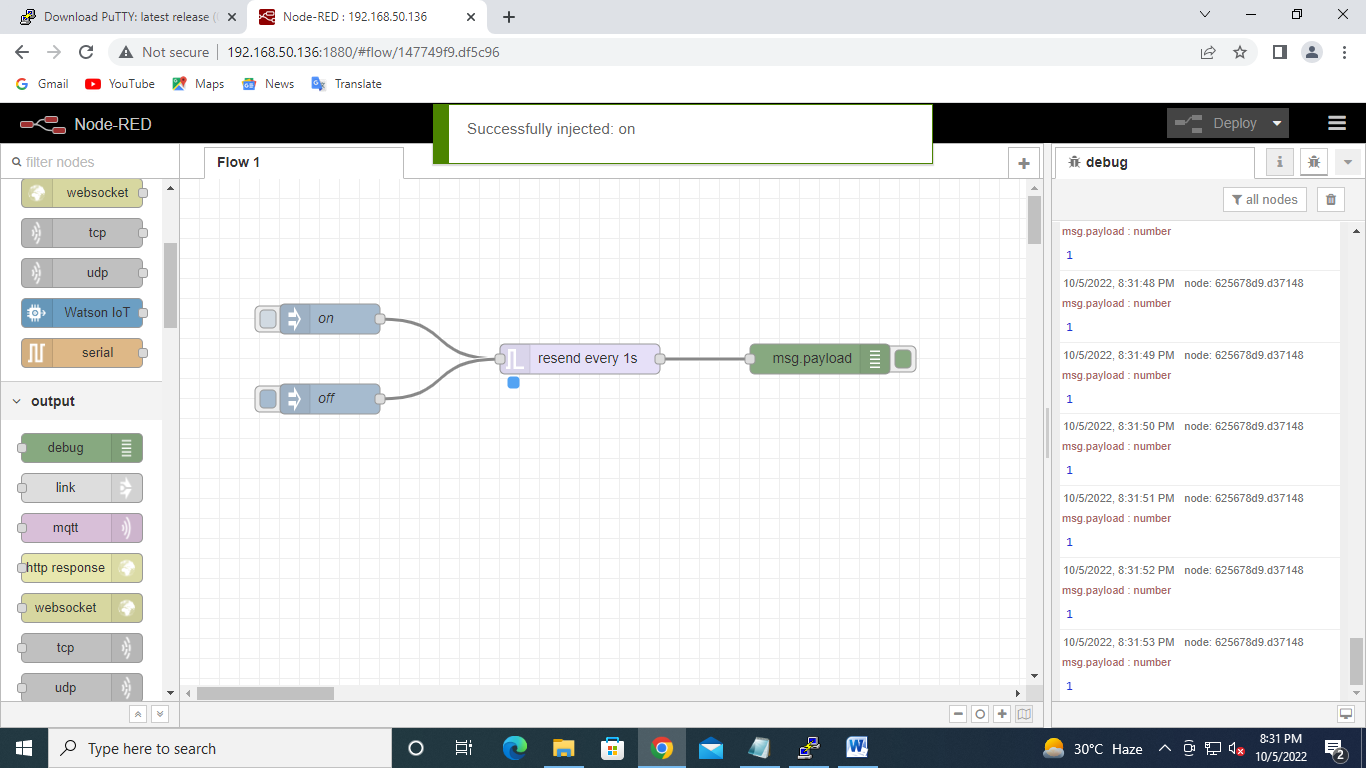


* Double click on trigger the node properties will be displayed.
* In send property, click on the dropdown and select **the existing msg object** option**.**
* In then attribute, click on the drop down and select **resend it every** option.
* Select **seconds** instead of Milliseconds and type **1** instead of 250.
* Click on Done and deploy it directly.



* Click on the ON and OFF button which is present beside the respective node.
* The output will be displayed in the debug section where until and unless you click on message payload button to deactivate the messages, it will keep on displaying new messages after every second.

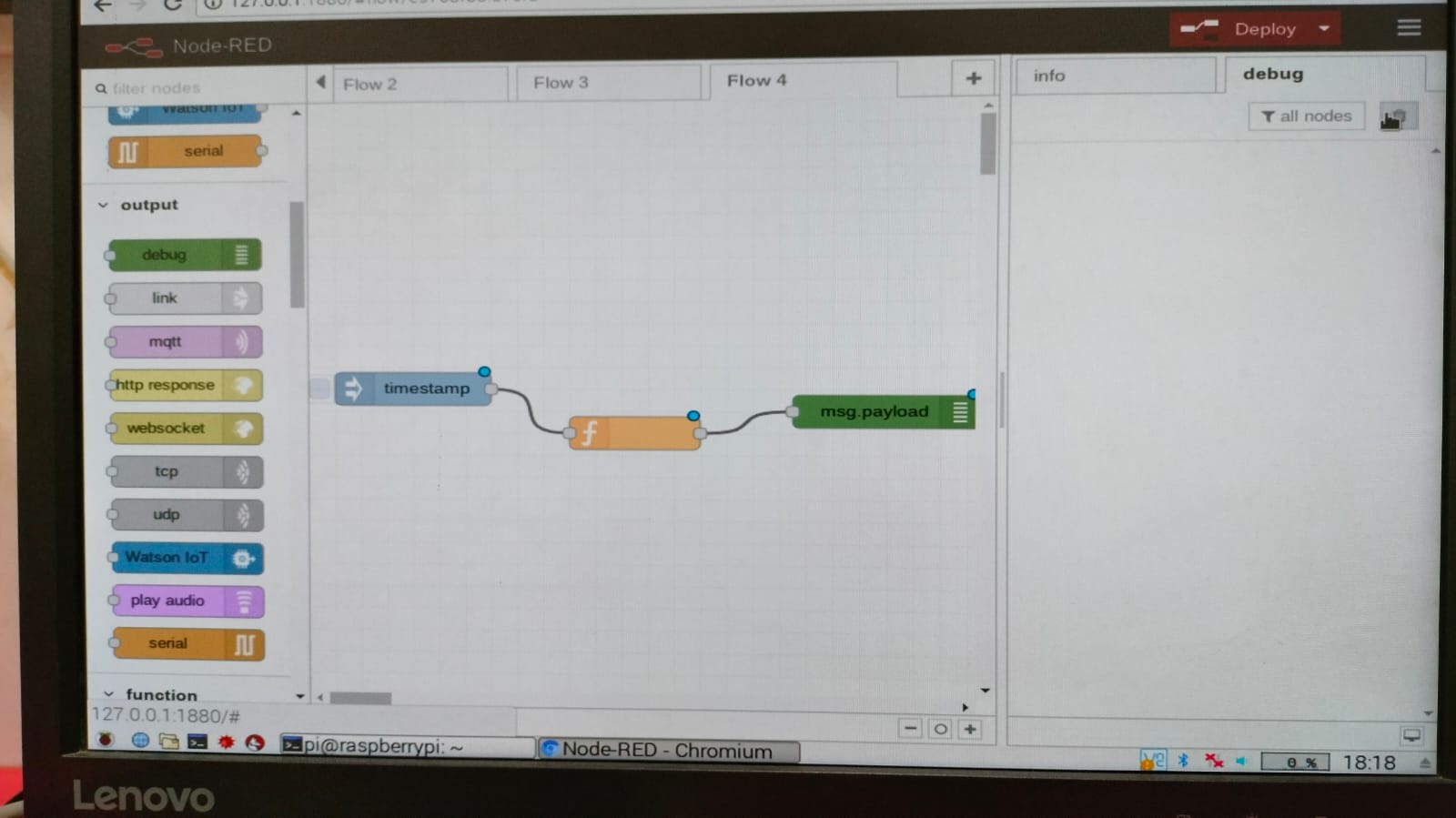




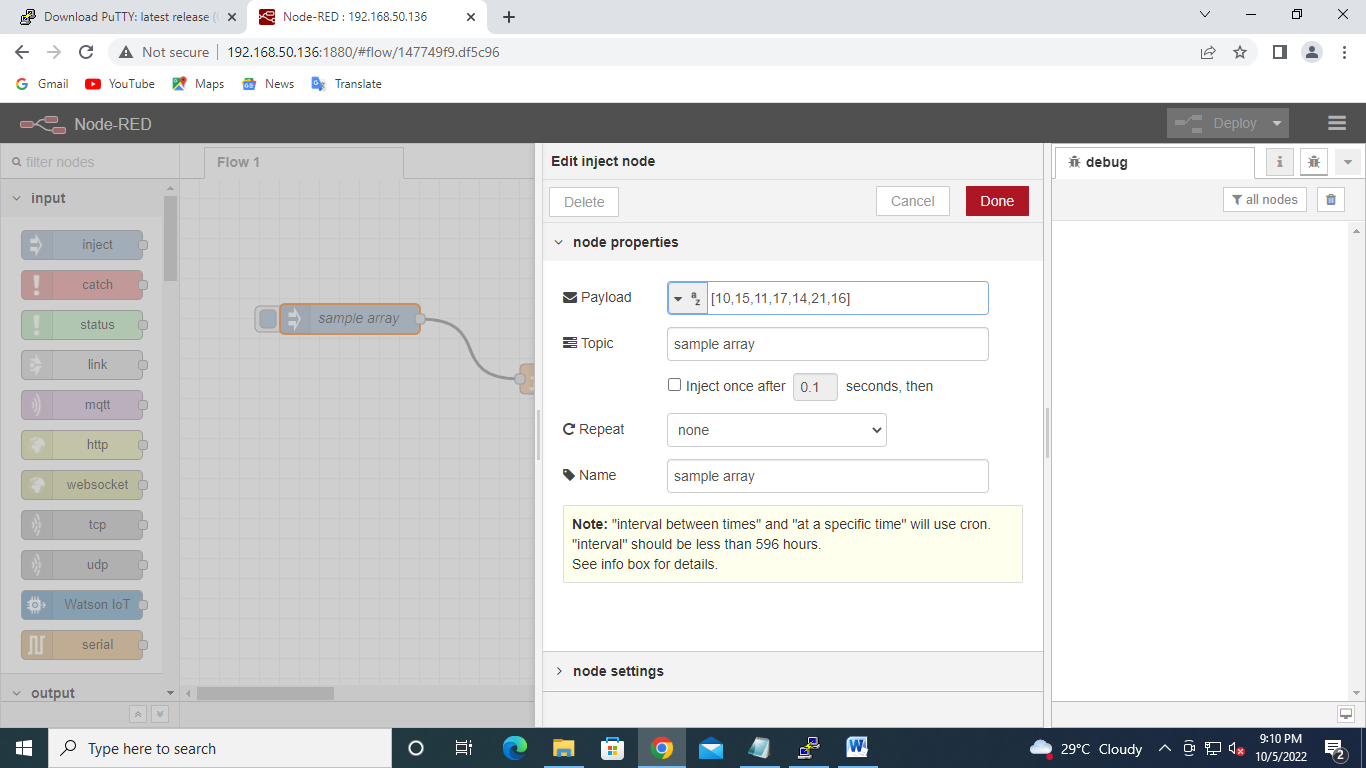
**PRACTICAL NO-5**

**Write simple JavaScript functions in Node-Red.**

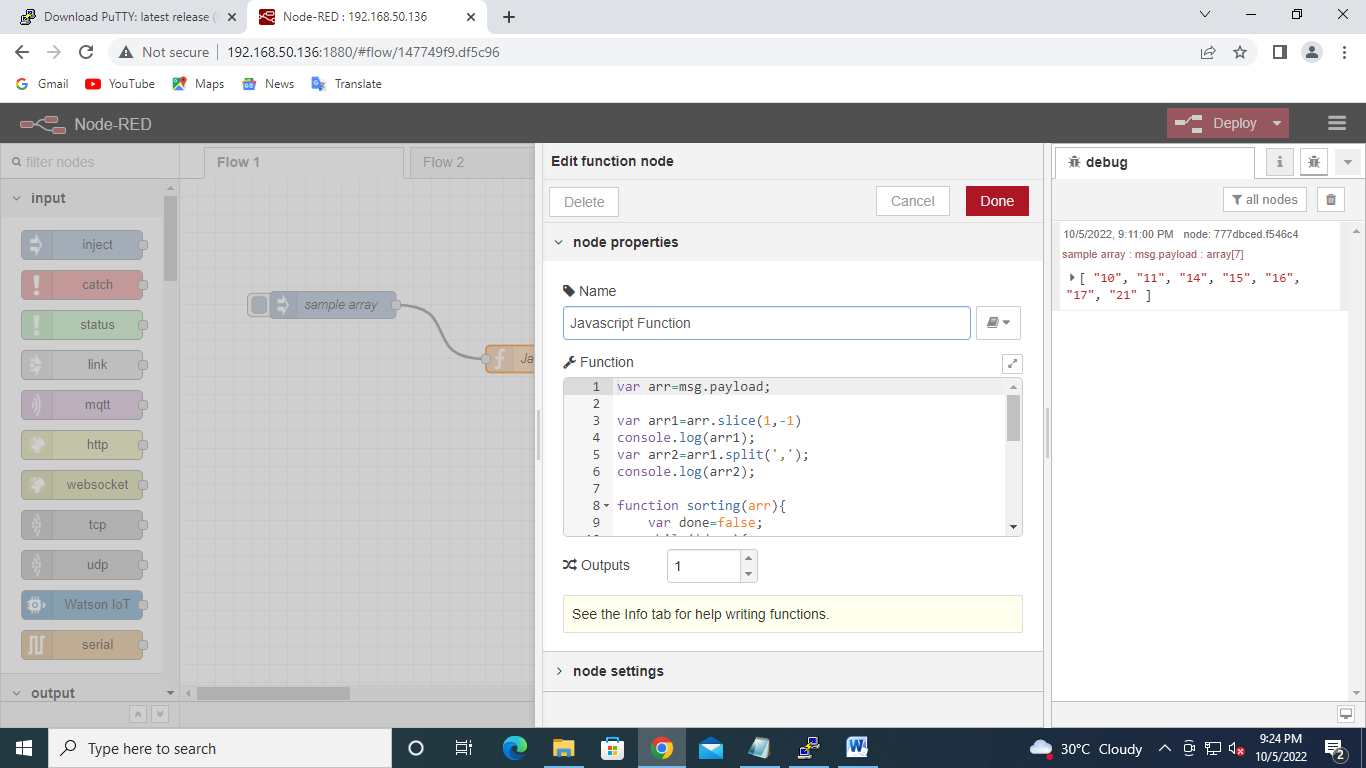
* Drag and drop **one inject** **node** from the input section, **one debug** from the output section **and one function** fromthe left side panel.
* Make a connection accordingly as per the given diagram below



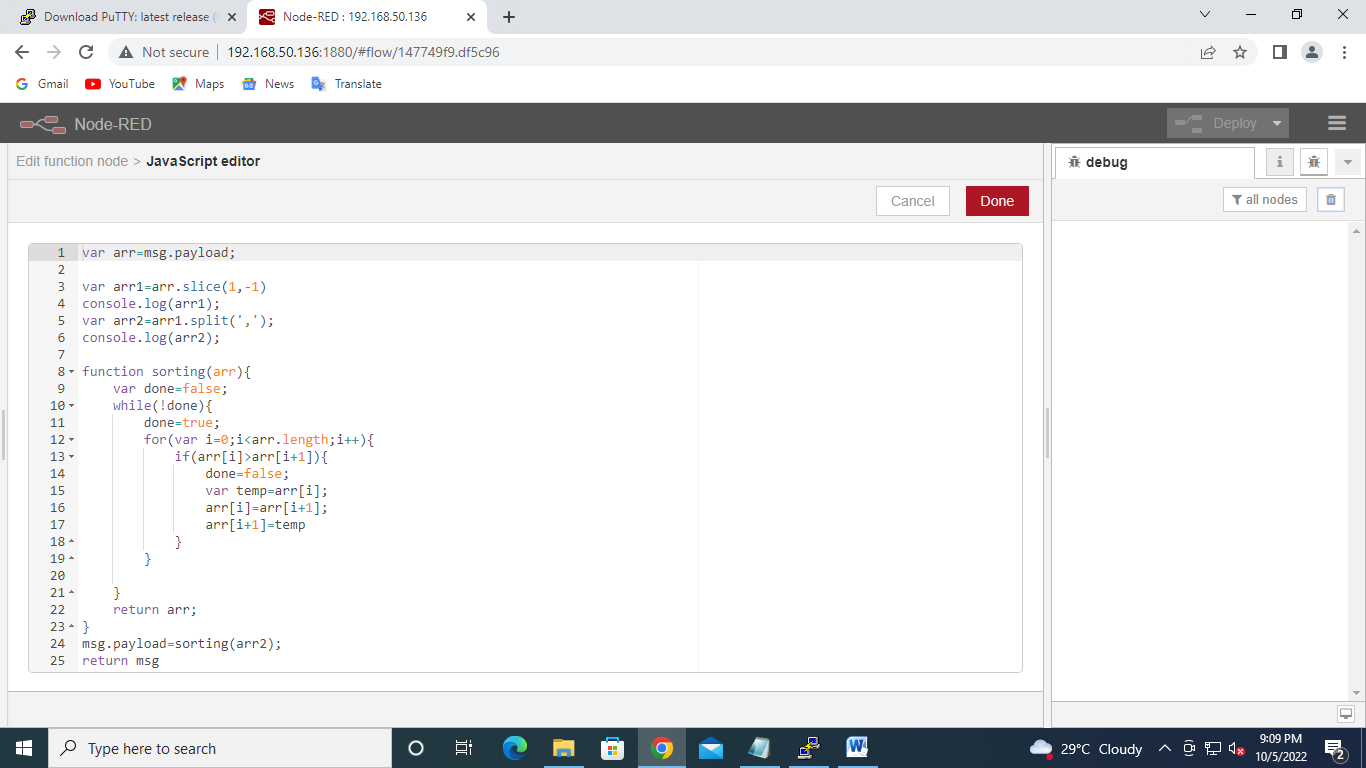
* Double click on the inject node, it will display the properties of that node.
* In payload, click on the dropdown and select **string** and type **[10,15,11,17,14,21,16]** in the beside input and in name section type **Sample Array**. Then click on Done.

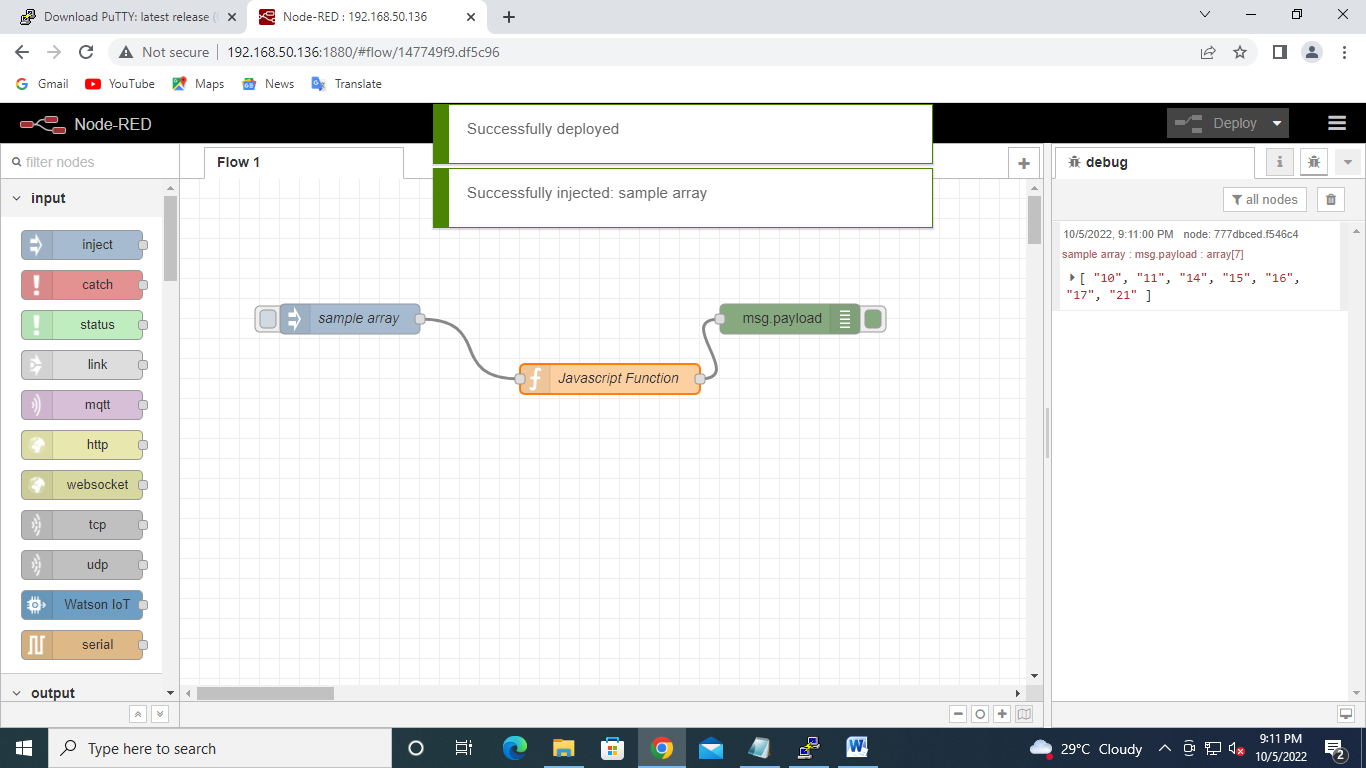


* Now double click on function, in the name input type **JavaScript Function.**
* Click on the function and start typing the code. The code is available below.
* After that click on Done and then click on Deploy.



* A Simple Code of JavaScript Function.





**PRACTICAL NO-6**

**Trigger a set of led Gpios on the pi via a Python Flask web server.**

* Type the below command in the Raspberry pi terminal

pi@raspberrypi ~ $ sudo pip install flask

* Creating the Python Script

This is the core script of our application. It sets up the web server and interacts with the Raspberry Pi GPIOs.

* To keep everything organized, start by creating a new folder:

pi@raspberrypi ~ $ mkdir web-server

pi@raspberrypi ~ $ cd web-server

pi@raspberrypi:~/web-server $

* Create a new file called app.py.

pi@raspberrypi:~/web-server $ nano app.py

* Type the following script(code) to your Raspberry Pi

'''

Adapted excerpt from Getting Started with Raspberry Pi by Matt Richardson

Modified by Rui Santos

Complete project details: http://randomnerdtutorials.com

'''

import RPi.GPIO as GPIO

from flask import Flask, render\_template, request

app = Flask(\_\_name\_\_)

GPIO.setmode(GPIO.BCM)

# Create a dictionary called pins to store the pin number, name, and pin state:

pins = {

20 : {'name' : 'GPIO 20', 'state' : GPIO.LOW},

21 : {'name' : 'GPIO 21', 'state' : GPIO.LOW}

}

# Set each pin as an output and make it low:

for pin in pins:

GPIO.setup(pin, GPIO.OUT)

GPIO.output(pin, GPIO.LOW)

@app.route("/")

def main():

# For each pin, read the pin state and store it in the pins dictionary:

for pin in pins:

pins[pin]['state'] = GPIO.input(pin)

# Put the pin dictionary into the template data dictionary:

templateData = {

'pins' : pins

}

# Pass the template data into the template main.html and return it to the user

return render\_template('main.html', \*\*templateData)

# The function below is executed when someone requests a URL with the pin number and action in it:

@app.route("/<changePin>/<action>")

def action(changePin, action):

# Convert the pin from the URL into an integer:

changePin = int(changePin)

# Get the device name for the pin being changed:

deviceName = pins[changePin]['name']

# If the action part of the URL is "on," execute the code indented below:

if action == "on":

# Set the pin high:

GPIO.output(changePin, GPIO.HIGH)

# Save the status message to be passed into the template:

message = "Turned " + deviceName + " on."

if action == "off":

GPIO.output(changePin, GPIO.LOW)

message = "Turned " + deviceName + " off."

# For each pin, read the pin state and store it in the pins dictionary:

for pin in pins:

pins[pin]['state'] = GPIO.input(pin)

# Along with the pin dictionary, put the message into the template data dictionary:

templateData = {

'pins' : pins

}

return render\_template('main.html', \*\*templateData)

if \_\_name\_\_ == "\_\_main\_\_":

app.run(host='0.0.0.0', port=80, debug=True)

* Click Ctrl+X and then click on Y key
* Click on Ctrl+C and then click on Enter to save the code
* Creating the HTML File

Keeping HTML tags separated from your Python script is how you keep your project organized.

Flask uses a template engine called [Jinja2](http://jinja.pocoo.org/docs/templates/) that you can use to send dynamic data from your Python script to your HTML file.

* Create a new folder called templates:

pi@raspberrypi:~/web-server $ mkdir templates

pi@raspberrypi:~/web-server $ cd templates

pi@raspberrypi:~/web-server/templates $

* Create a new file called main.html.

pi@raspberrypi:~/web-server/templates $ nano main.html

* Type the following script(code) to your Raspberry Pi:

<!DOCTYPE html>

<head>

<title>RPi Web Server</title>

<!-- Latest compiled and minified CSS -->

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css" integrity="sha384-1q8mTJOASx8j1Au+a5WDVnPi2lkFfwwEAa8hDDdjZlpLegxhjVME1fgjWPGmkzs7" crossorigin="anonymous">

<!-- Optional theme -->

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap-theme.min.css" integrity="sha384-fLW2N01lMqjakBkx3l/M9EahuwpSfeNvV63J5ezn3uZzapT0u7EYsXMjQV+0En5r" crossorigin="anonymous">

<!-- Latest compiled and minified JavaScript -->

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js" integrity="sha384-0mSbJDEHialfmuBBQP6A4Qrprq5OVfW37PRR3j5ELqxss1yVqOtnepnHVP9aJ7xS" crossorigin="anonymous"></script>

</head>

<body>

<h1>RPi Web Server</h1>

{% for pin in pins %}

<h2>{{ pins[pin].name }}

{% if pins[pin].state == true %}

is currently <strong>on</strong></h2><div class="row"><div class="col-md-2">

<a href="/{{pin}}/off" class="btn btn-block btn-lg btn-default" role="button">Turn off</a></div></div>

{% else %}

is currently <strong>off</strong></h2><div class="row"><div class="col-md-2">

<a href="/{{pin}}/on" class="btn btn-block btn-lg btn-primary" role="button">Turn on</a></div></div>

{% endif %}

{% endfor %}

</body>

</html>

* Launching the Web Server

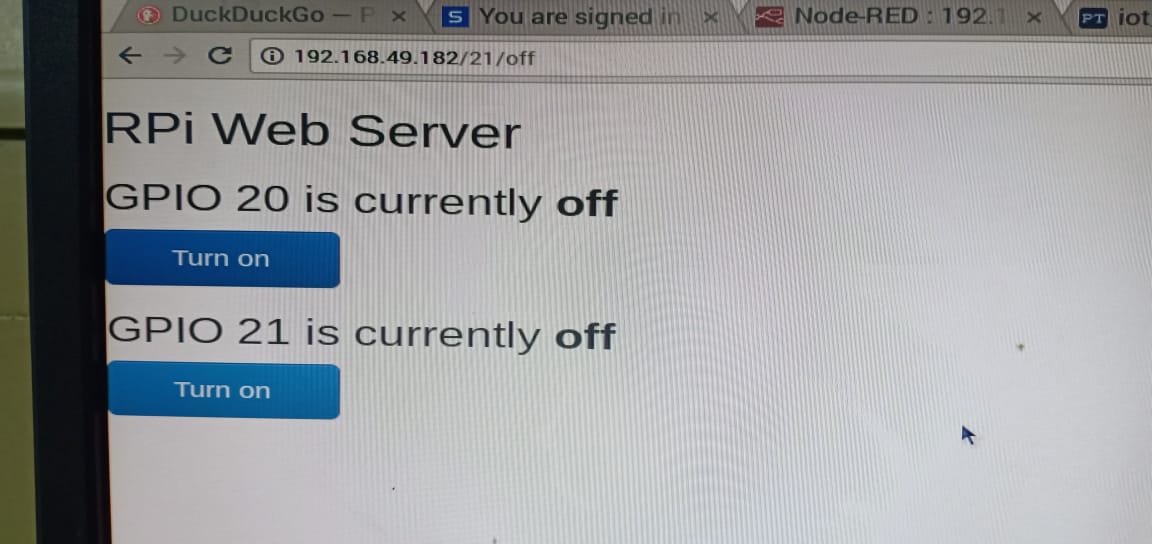
To launch your Raspberry Pi web server move to the folder that contains the file app.py:

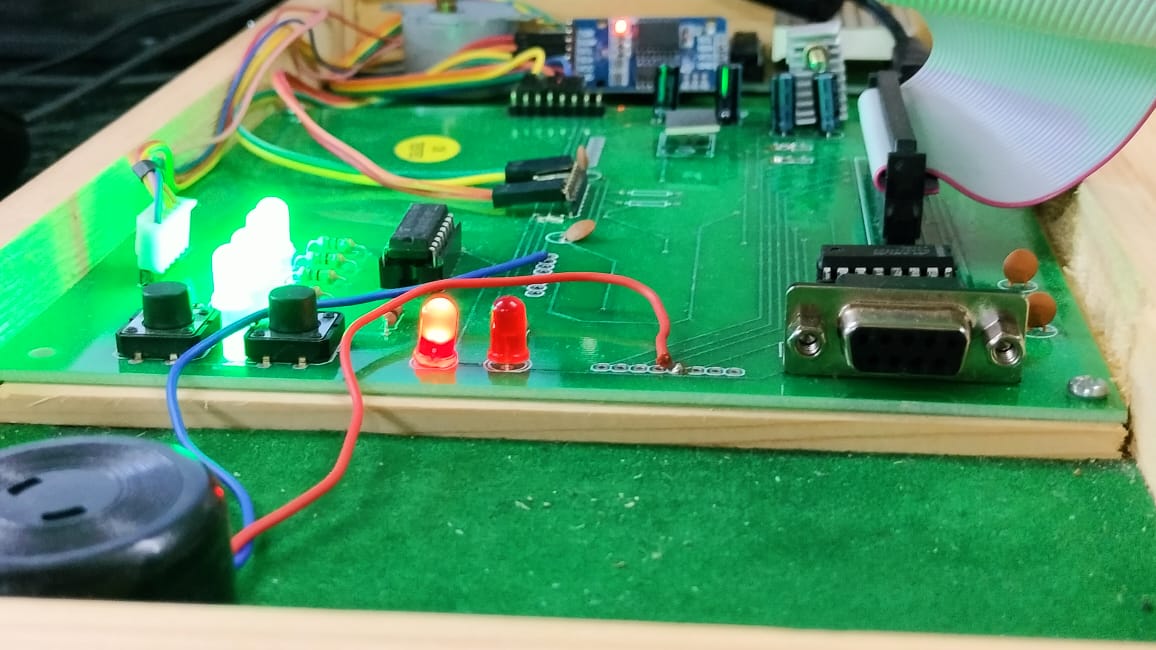
pi@raspberrypi:~/web-server/templates $ cd ..

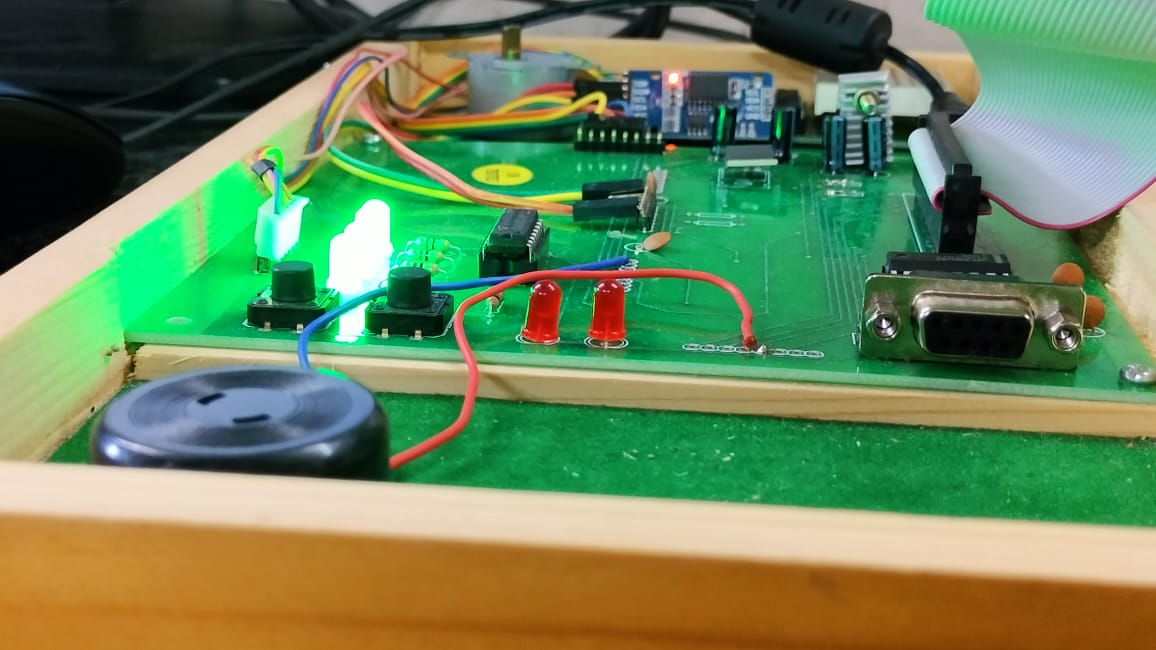
Then run the following command:

pi@raspberrypi:~/web-server $ sudo python app.py

Your web server should start immediately!







**PRACTICAL NO-7**

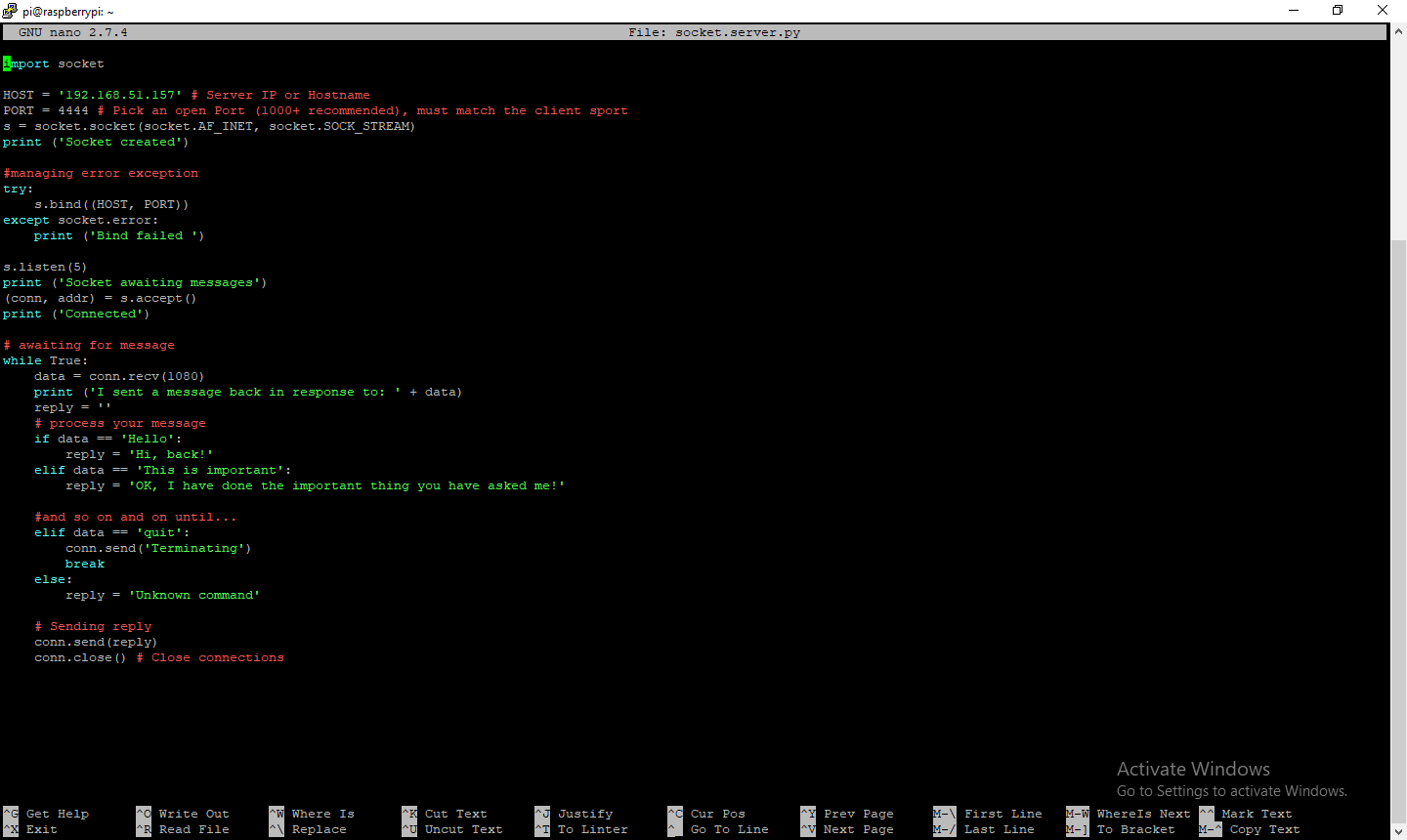
**Setup a TCP server and client on a raspberry pi using Python modules to send messages and execute shell commands from within python such as starting another application.**

* Open the terminal then type the command

**nano socket.server.py**

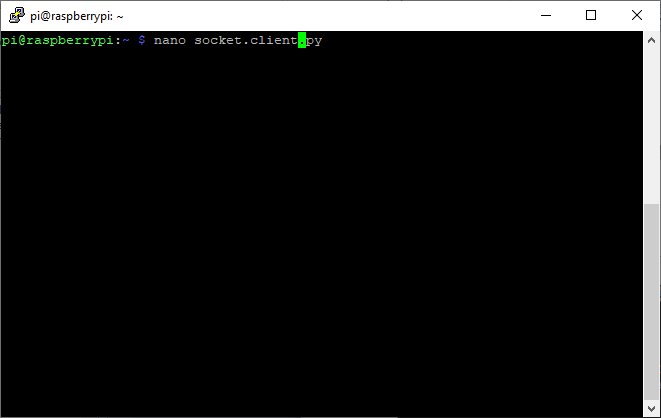


* Type the Code give below and Click Ctrl+X and then click on Y key
* Click on Ctrl+C and then click on Enter to save the code.

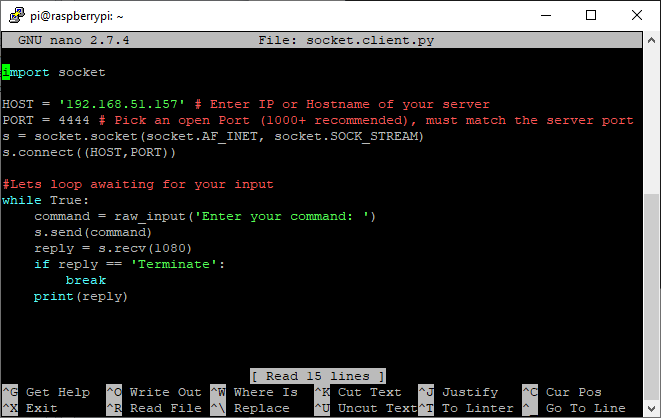


* Open the terminal then type the command

**nano socket.client.py**

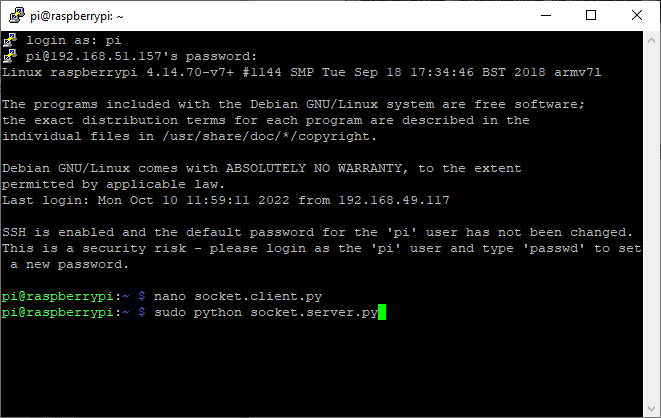


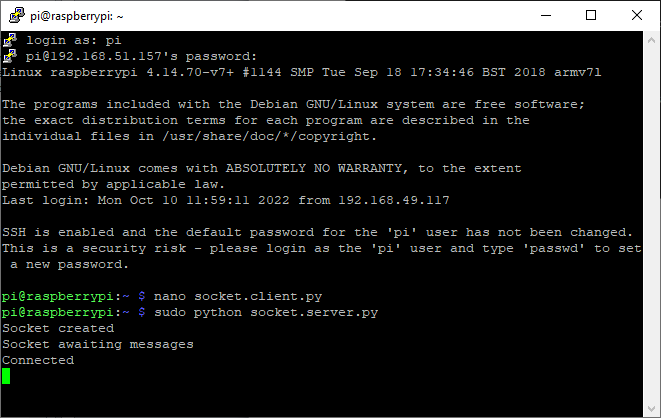
* Type the Code give below and Click Ctrl+X and then click on Y key
* Click on Ctrl+C and then click on Enter to save the code.



* Follow the below commands and type that in the terminal of the Raspberry pi.

sudo python socket.server.py





* Follow the below commands and type that in the windows terminal with the use of putty

sudo python socket.client.py

* Type Hello in the Command

**OUTPUT-**

