

Parle Tilak Vidyalaya Association's

# MULUND COLLEGE OF COMMERCE

DEPARTMENT OF COMPUTER SCIENCE

# IOT PRACTICALS & JOURNAL

## **Table of Contents**

- 1. Introduction to Raspberry Pi**
- 2. Linux Commands: Exploring the Raspbian**
- 3. GPIO: Light the LED with Python**
- 4. GPIO: LED Grid Module: Program the 8X8 Grid with Different Formulas**
- 5. SPI: Camera Connection and capturing Images using SPI**
- 6. Programs using PWM.**
- 7. Node RED: Connect LED to Internet of Things.**
- 8. Raspberry Pi Programming Using Online Simulator .**
- 9. Study Of XMPP Server Protocol .**
- 10. Study Of MQTT Protocol .**

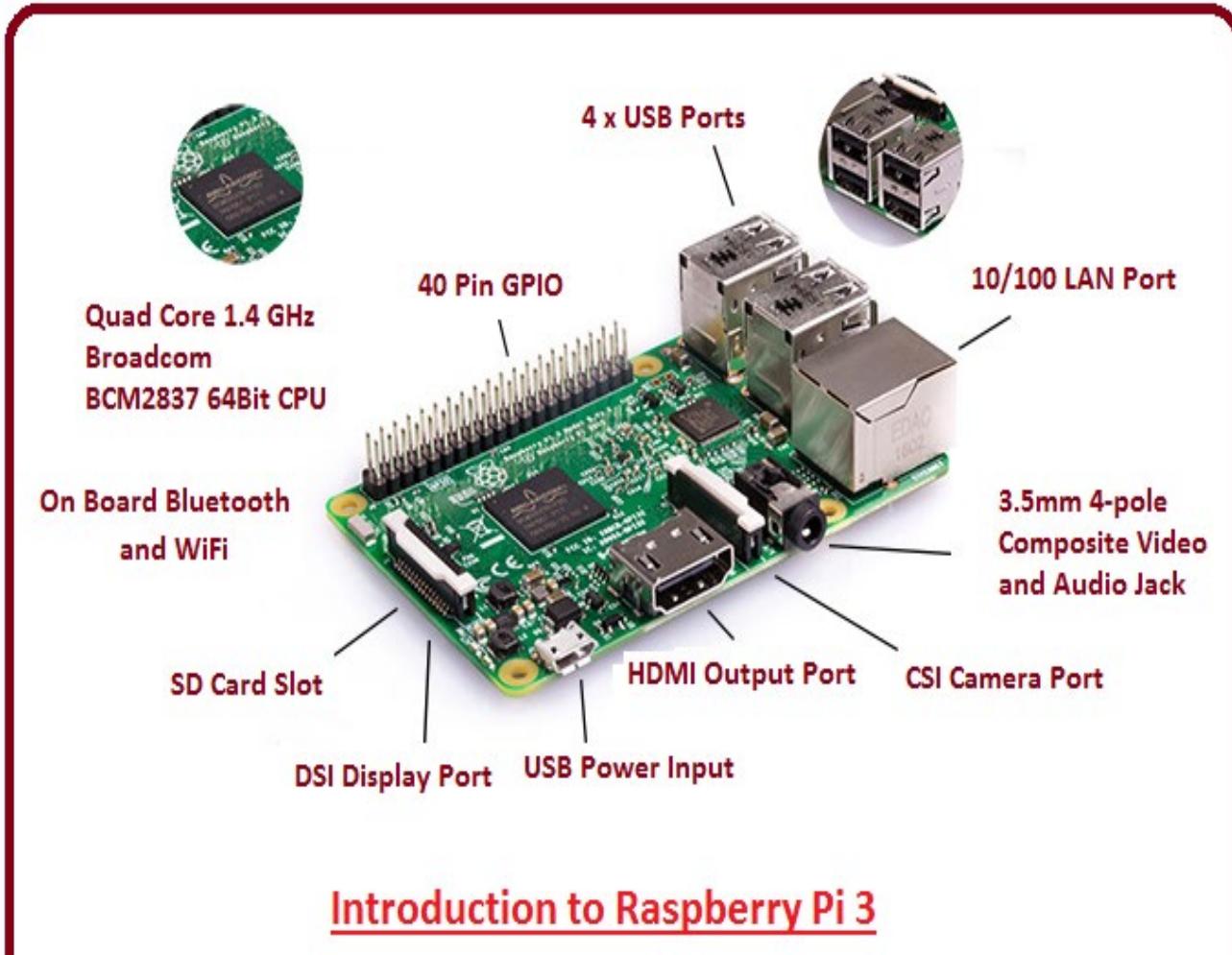
## PRACTICAL NO . 1

### INTRODUCTION TO RASPBERRY PI

This is an introduction of Raspberry Pi 3 Model B. We will explore what Raspberry Pi 3 has to offer in terms of its features and performance. Raspberry Pi as a world's most inexpensive and powerful Single Board Computer. Ever since the launch of Raspberry Pi from 2012, we have seen several version of it. This is world's cheapest microprocessor unit specially built for learner and makers. We can easily learn how software and hardware work together without been worrying about damage/cost. We can buy Raspberry Pi board with just somewhere around 35\$. The cost of Pi allows newbies to celebrate mistakes and learn most out of it. Also Raspberry Pi has a huge community and plenty of online resources which make learning smooth.

- **Raspberry Pi 3 Model B Features :** CPU: Raspberry Pi 3 uses Broadcom BCM2837 SOC 64-bit quad-core ARM Cortex A53 (ARMv8 CPU) with 512KB shared L2 cache.
- **Memory :** Provided with 1 GB of RAM Wi-Fi Support: 802.11n .
- **Wireless LAN Bluetooth:** Supports Bluetooth 4.1 Bluetooth Low Energy (BLE).
- **USB Ports :** 4-USB ports which allow attaching four different USB devices like keyboard, mouse, etc.
- **Ethernet Port :** Standard Ethernet port to quickly setup and access internet. This can be very useful when we want to setup raspberry pi for the first time without a monitor.
- **GPIO Pins:** Raspberry Pi 3 supports 40 GPIO Pins General Purpose Input Output. These digital input/output pins can be used to drive LED, Switches, and Sensors etc.
- **Full HDMI Port:** Support HDMI port (High-Definition Multimedia Interface) which can be used to quickly connect raspberry pi to HDMI Monitor. With HDMI Cable and Monitor we can add Screen to Raspberry Pi.
- **Micro SD card slot:** The Micro SD Card will hold the operating system which will boot while we power on Raspberry Pi 3. In next tutorial, we will learn how to setup and prepare SD card with Raspbian OS.
- **Audio/Video:** Combined 3.5mm audio jack and composite video .
- **Display interface (DSI):** enable us to interface Display Module .
- **Camera interface (CSI):** enable us to interface Camera Module

- **Graphics Support:** VideoCore IV 3D graphics core for advance graphics capabilities.





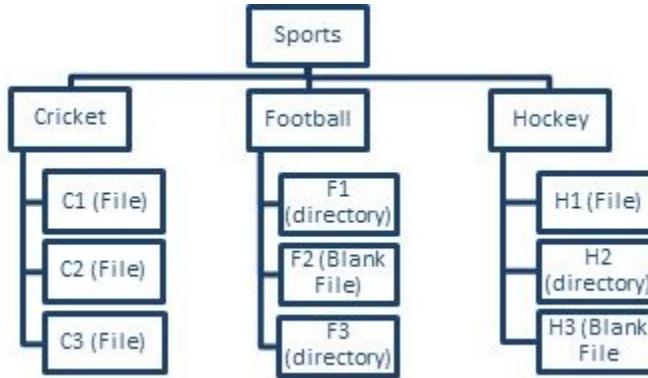
Alternate Function	
3.3V PWR	1
I2C1 SDA	GPIO 2
I2C1 SCL	GPIO 3
GPIO 4	5
GND	9
GPIO 17	11
GPIO 27	13
GPIO 22	15
3.3V PWR	17
SPI0 MOSI	GPIO 10
SPI0 MISO	GPIO 9
SPI0 SCLK	GPIO 11
GND	25
Reserved	27
GPIO 5	29
GPIO 6	31
GPIO 13	33
SPI1 MISO	GPIO 19
GPIO 26	37
GND	39
5V PWR	2
5V PWR	4
GND	6
UART0 TX	8
UART0 RX	10
GPIO 18	12
GND	14
GPIO 23	16
GPIO 24	18
GND	20
GPIO 25	22
GPIO 8	24
GPIO 7	26
Reserved	28
GND	30
GPIO 12	32
GND	34
GPIO 16	36
GPIO 20	38
SPI1 MOSI	39
SPI1 CS0	40
SPI1 SCLK	41

## PRACTICAL NO.2

### Linux Commands: Exploring the Raspbian

Q1].

- a. Create a structure as given below by using Linux.



```
File Edit Tabs Help
pi@raspberrypi:~ $ mkdir sport
pi@raspberrypi:~ $ cd sport
pi@raspberrypi:~/sport $ mkdir cricket
pi@raspberrypi:~/sport $ mkdir football
pi@raspberrypi:~/sport $ mkdir hockey\
> ;
pi@raspberrypi:~/sport $ mkdir hockey
mkdir: cannot create directory 'hockey': File exists
pi@raspberrypi:~/sport $ cd cricket
pi@raspberrypi:~/sport/cricket $ cat<C1>
bash: syntax error near unexpected token `newline'
pi@raspberrypi:~/sport/cricket $ cat <C1>
bash: syntax error near unexpected token `newline'
pi@raspberrypi:~/sport/cricket $ cat<C1>
bash: C1: No such file or directory
pi@raspberrypi:~/sport/cricket $ cat>C1
abcd
heman pawar
topper
vikram patil
pi@raspberrypi:~/sport/cricket $ cat>C2
prasad
pawar playboy
pi@raspberrypi:~/sport/cricket $ cat>C3
Heman is a very good boy.
He is the best
pi@raspberrypi:~/sport/cricket $ cd football
bash: cd: football: No such file or directory
pi@raspberrypi:~/sport/cricket $ cd..
bash: cd..: command not found
pi@raspberrypi:~/sport/cricket $ cd ..
pi@raspberrypi:~/sport $ cd football
pi@raspberrypi:~/sport/football $ cat>ft1
pi@raspberrypi:~/sport/football $ touch ft1
pi@raspberrypi:~/sport/football $ mkdir ft2
pi@raspberrypi:~/sport/football $ mkdir ft3
pi@raspberrypi:~/sport/football $ cd ..
pi@raspberrypi:~/sport $ cd hockey
pi@raspberrypi:~/sport/hockey $ cat>h1
Hockey is a very good sport
I am a disco dancer
pi@raspberrypi:~/sport/hockey $ touch h3
pi@raspberrypi:~/sport/hockey $ mkdir h2
pi@raspberrypi:~/sport/hockey $ cd ..
```

- b. Display the complete structure of Sports directory.
- c. Copy the file C2 from cricket to F3.

```

pi@raspberrypi:~/sport/football $ mkdir ft3
pi@raspberrypi:~/sport/football $ cd ..
pi@raspberrypi:~/sport $ cd hockey
pi@raspberrypi:~/sport/hockey $ cat>h1
Hockey is a very good sport
I am a disco dancer
pi@raspberrypi:~/sport/hockey $ touch h3
pi@raspberrypi:~/sport/hockey $ mkdir h2
pi@raspberrypi:~/sport/hockey $ cd ..
pi@raspberrypi:~/sport $ cd ..
pi@raspberrypi:~ $ tree sport
sport
├── cricket
│   ├── C1
│   ├── C2
│   └── C3
├── football
│   ├── ft1
│   ├── ft2
│   └── ft3
└── hockey
    ├── h1
    ├── h2
    └── h3

6 directories, 6 files
pi@raspberrypi:~ $ cp -r sport/cricket/C2 sport/football/ft3
pi@raspberrypi:~ $ tree sport
sport
├── cricket
│   ├── C1
│   ├── C2
│   └── C3
├── football
│   ├── ft1
│   ├── ft2
│   └── ft3
└── hockey
    ├── h1
    ├── h2
    └── h3

6 directories, 7 files

```

Q2].

- a. Display the help file for the command pwd.
- b. Remove the file H1.

```

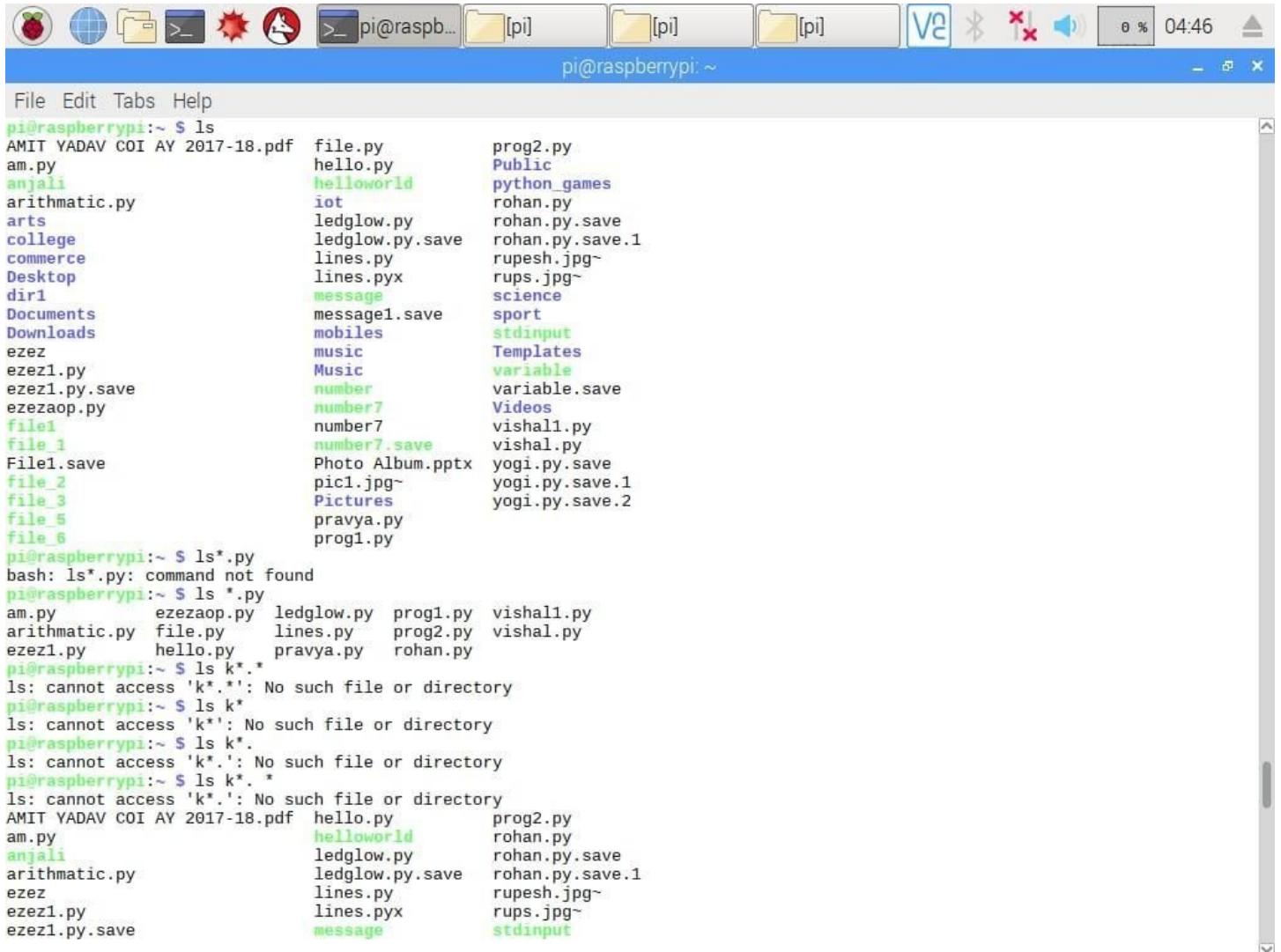
pi@raspberrypi:~ $ man pwd
pi@raspberrypi:~ $ cd sport
pi@raspberrypi:~/sport $ cd hockey
pi@raspberrypi:~/sport/hockey $ rm h1
pi@raspberrypi:~/sport/hockey $ cd ..
pi@raspberrypi:~/sport $ cd ..
pi@raspberrypi:~ $ tree sport
sport
├── cricket
│   ├── C1
│   ├── C2
│   └── C3
├── football
│   ├── ft1
│   ├── ft2
│   └── ft3
└── hockey
    ├── h2
    └── h3

6 directories, 6 files

```

- c. List all the files present under root directory .(\$ ls)
- d. Display all python files.(\$ ls +.py)

- e. List all the files which starts with letter 'k'.(\$ ls k\*\*)  
f. Display content of file C2.



```

pi@raspberrypi:~ $ ls
AMIT YADAV COI AY 2017-18.pdf  file.py      prog2.py
am.py                      hello.py     Public
anjali                     helloworld  python_games
arithmatic.py               iot          rohan.py
arts                       ledglow.py   rohan.py.save
college                    ledglow.py.save rohan.py.save.1
commerce                   lines.py    rupesh.jpg-
Desktop                   lines.pyx   rups.jpg~
dir1                      message     science
Documents                  message1.save sport
Downloads                 mobiles      stdinput
ezez                      music       Templates
ezez1.py                  Music       variable
ezez1.py.save              number     Videos
ezezaop.py                number7    vishal1.py
file1                     number8    vishal.py
file_1                    number7.save yogi.py.save
File1.save                 Photo Album.pptx  yogi.py.save.1
file_2                     pic1.jpg~  Pictures   yogi.py.save.2
file_3                     Pictures
file_5                     pravya.py
file_6                     prog1.py

pi@raspberrypi:~ $ ls*.py
bash: ls*.py: command not found
pi@raspberrypi:~ $ ls *.py
am.py          ezezaop.py  ledglow.py  prog1.py  vishal1.py
arithmatic.py  file.py    lines.py    prog2.py  vishal.py
ezez1.py      hello.py   pravya.py  rohan.py

pi@raspberrypi:~ $ ls k*.*
ls: cannot access 'k*.*': No such file or directory
pi@raspberrypi:~ $ ls k*
ls: cannot access 'k*': No such file or directory
pi@raspberrypi:~ $ ls k*.
ls: cannot access 'k*.': No such file or directory
pi@raspberrypi:~ $ ls k*.*
ls: cannot access 'k*.*': No such file or directory
AMIT YADAV COI AY 2017-18.pdf  hello.py      prog2.py
am.py                      helloworld  rohan.py
anjali                     ledglow.py   rohan.py.save
arithmatic.py               ledglow.py.save rohan.py.save.1
ezez                      lines.py    rupesh.jpg-
ezez1.py                  lines.pyx   rups.jpg~
ezez1.py.save              message     stdinput

```

```
File Edit Tabs Help
catgirl.png inkspillsettings.png star_title.png
cat.png inkspillspot.png tetrisb.mid
drawing.py launcher.sh tetrisc.mid
flippybackground.png match0.wav tetrominoforidiots.py
flippyboard.png match1.wav tetromino.py
flippy.py match2.wav Tree_Short.png
fourinarow.py match3.wav Tree_Tall.png
gameicon.png match4.wav Tree_Ugly.png
gem1.png match5.wav Wall_Block_Tall.png
gem2.png memoriypuzzle_obfuscated.py Wood_Block_Tall.png
gem3.png memoriypuzzle.py wormy.py

science:
f1 f2 f3

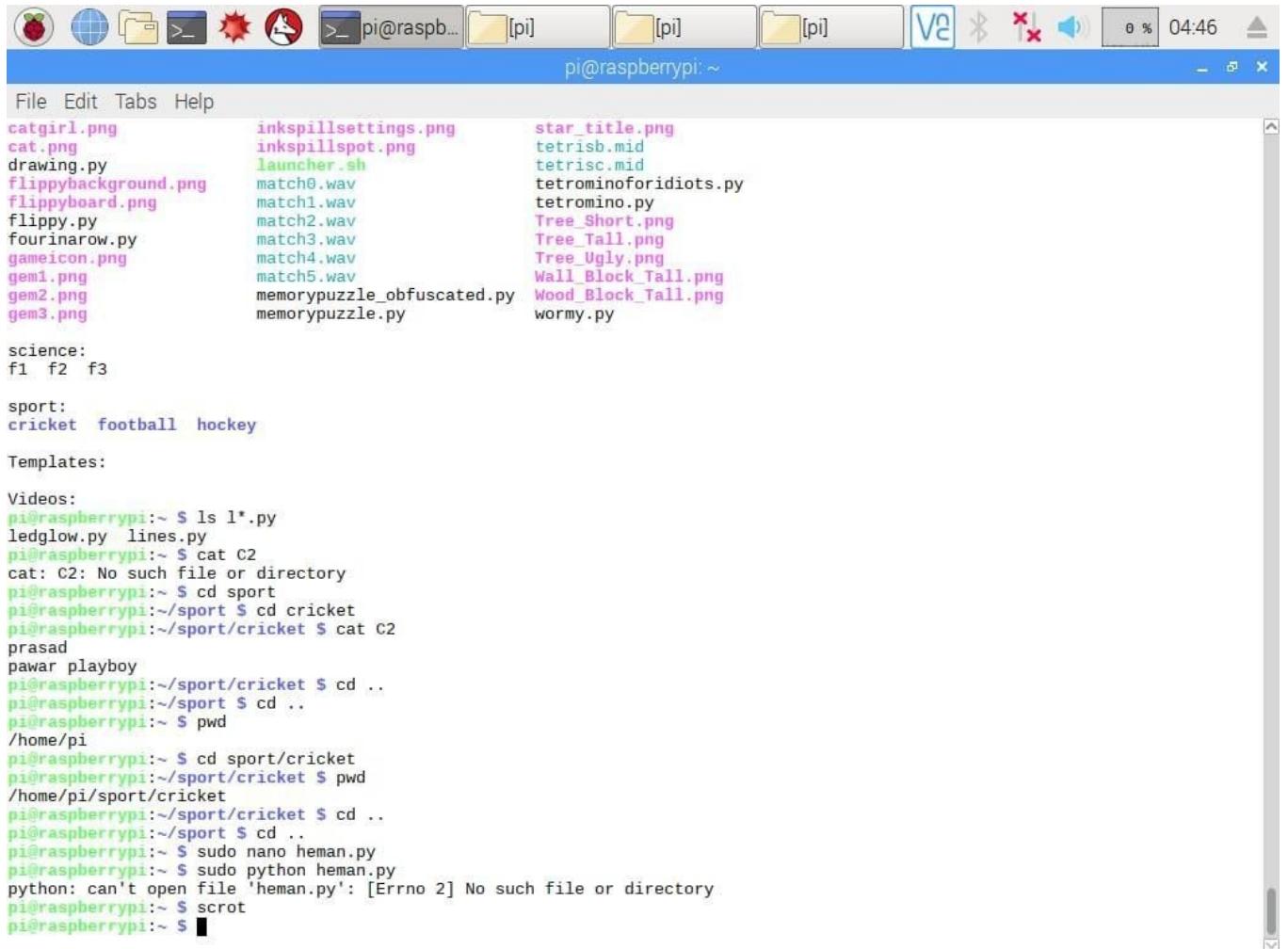
sport:
cricket football hockey

Templates:

Videos:
pi@raspherrypi:~ $ ls l*.py
ledglow.py lines.py
pi@raspherrypi:~ $ cat C2
cat: C2: No such file or directory
pi@raspherrypi:~ $ cd sport
pi@raspherrypi:~/sport $ cd cricket
pi@raspherrypi:~/sport/cricket $ cat C2
prasad
pawar playboy
pi@raspherrypi:~/sport/cricket $ cd ..
pi@raspherrypi:~/sport $ cd ..
pi@raspherrypi:~ $ pwd
/home/pi
pi@raspherrypi:~ $ cd sport/cricket
pi@raspherrypi:~/sport/cricket $ pwd
/home/pi/sport/cricket
pi@raspherrypi:~/sport/cricket $ cd ..
pi@raspherrypi:~/sport $ cd ..
pi@raspherrypi:~ $ sudo nano heman.py
pi@raspherrypi:~ $ sudo python heman.py
python: can't open file 'heman.py': [Errno 2] No such file or directory
pi@raspherrypi:~ $ scrot
pi@raspherrypi:~ $
```

Q3]. Write the commands :

- Display pwd.
- Write the steps to create a python file.
- Write the shortcut for saving a python file, exit the window, come back to the terminal.
- How to execute a python file ?



The screenshot shows a Raspberry Pi desktop environment with a terminal window open. The terminal window title is "pi@raspberrypi: ~". The desktop icons include a pi logo, a globe, a folder, a terminal icon, a red star, a dog icon, and three folder icons labeled "[pi]". The terminal window displays the following text:

```
File Edit Tabs Help
catgirl.png      inkspillsettings.png    star_title.png
cat.png          inkspillspot.png       tetrisb.mid
drawing.py        launcher.sh           tetrisc.mid
flippybackground.png  match0.wav       tетrominoforidiots.py
flippyboard.png   match1.wav       tетromino.py
flippy.py         match2.wav       Tree_Short.png
fourinarow.py     match3.wav       Tree_Tall.png
gameicon.png      match4.wav       Tree_Ugly.png
gem1.png          match5.wav       Wall_Block_Tall.png
gem2.png          memorypuzzle_obfuscated.py Wood_Block_Tall.png
gem3.png          memorypuzzle.py    wormy.py

science:
f1 f2 f3

sport:
cricket football hockey

Templates:

Videos:
pi@raspberrypi:~ $ ls l*.py
ledglow.py lines.py
pi@raspberrypi:~ $ cat C2
cat: C2: No such file or directory
pi@raspberrypi:~ $ cd sport
pi@raspberrypi:~/sport $ cd cricket
pi@raspberrypi:~/sport/cricket $ cat C2
prasad
pawar playboy
pi@raspberrypi:~/sport/cricket $ cd ..
pi@raspberrypi:~/sport $ cd ..
pi@raspberrypi:~ $ pwd
/home/pi
pi@raspberrypi:~ $ cd sport/cricket
pi@raspberrypi:~/sport/cricket $ pwd
/home/pi/sport/cricket
pi@raspberrypi:~/sport/cricket $ cd ..
pi@raspberrypi:~/sport $ cd ..
pi@raspberrypi:~ $ sudo nano heman.py
pi@raspberrypi:~ $ sudo python heman.py
python: can't open file 'heman.py': [Errno 2] No such file or directory
pi@raspberrypi:~ $ scrot
pi@raspberrypi:~ $
```

## **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.
- 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.
- 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 NO. 3

### **GPIO: Light the LED with Python**

Once you've setup your Raspberry Pi according to the [getting started tutorial](#), you are ready for your first real project. Let's light up an led using the Python programming language and the GPIO pins on your Raspberry Pi, hereafter called **RPi**.

#### **What You Will Learn:**

- You will construct a basic electrical circuit and attach it to your RPi GPIO pins
- You will write a simple Python program to control the circuit using IDLE IDE

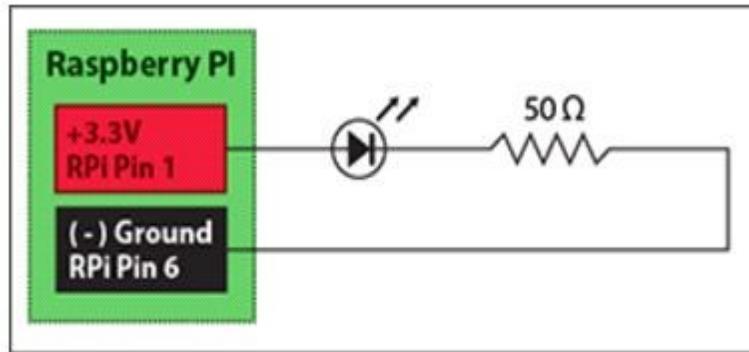
#### **What You Will Need:**

- Raspberry Pi configured with the [GPIO library](#)
- 1 - small led, any color
- 1 - 50 ohm resistor
- Some small-gauge solid core wire
- Breadboard and/or alligator clips to hold connections

#### **Let There Be Light**

Before we get around to writing any code, let's first get acquainted with the pin numbering of our RPi and create a simple circuit. We'll start by simply lighting our led using the 3.3v pin and the ground pin on our RPi. We will use

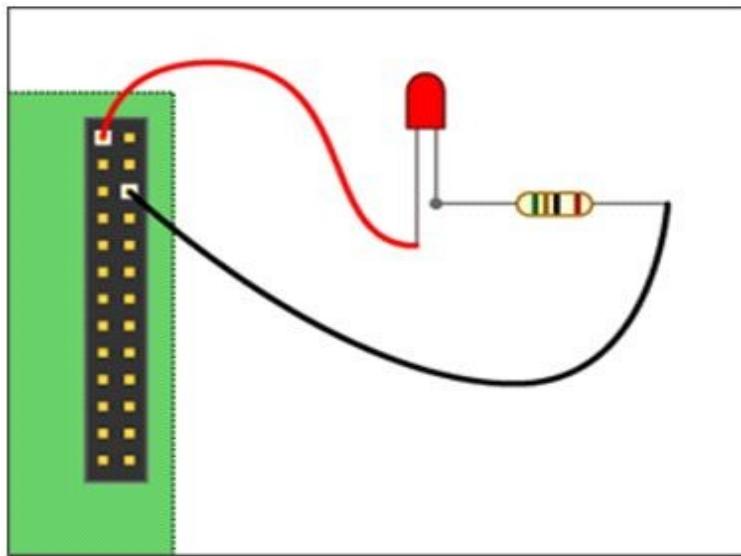
the following schematic for our circuit:



Before starting, unplug your RPi. You wouldn't want to risk shorting it out while working with it 'powered on', especially since this is our first project.

- Using your assortment of materials, create the circuit on either your breadboard, or using your alligator clips.
- Pin 1 (+3.3 volts) the longer leg of your led. This pin provides a steady supply of 3.3v. Unlike the GPIO pins on your RPi, this pin is not programmable, and cannot be controlled by software.
- Attach the shorter leg of the led to the resistor. Finally, attach the other end of the resistor to Pin 6 (- ground) on your RPi.

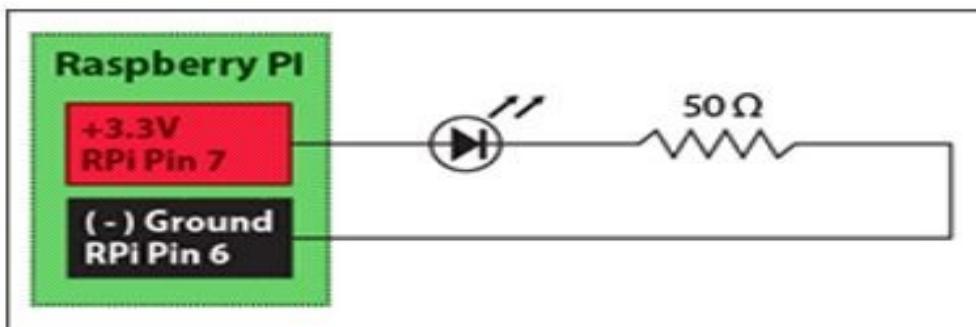
Double-check your connections. When you are done, your circuit should look like this:



- Power on your RPi - the led should immediately turn on.

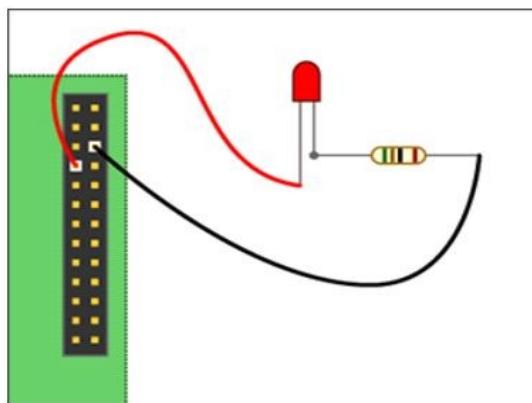
### Controlling The Led With Python :

Now that we've tested our basic circuit, it's time to move the positive lead from the 'always on' 3.3v pin to one of the programmable GPIO pins. Here is what our circuit will look like:



- Power-off your RPi again before making any changes to your wiring.
- Move the positive lead from pin 1 to pin 7.

When you are done, your circuit should look like this:



## Writing The Code

To write any python scripts using the IDLE IDE because it comes packaged with the Raspbian distribution, it's free, and it makes writing and debugging code a bit simpler than when using Python command line or a text editor. It's important to note that when writing python scripts that utilize the GPIO pins, you must run them as a **superuser** or your scripts will not run properly.

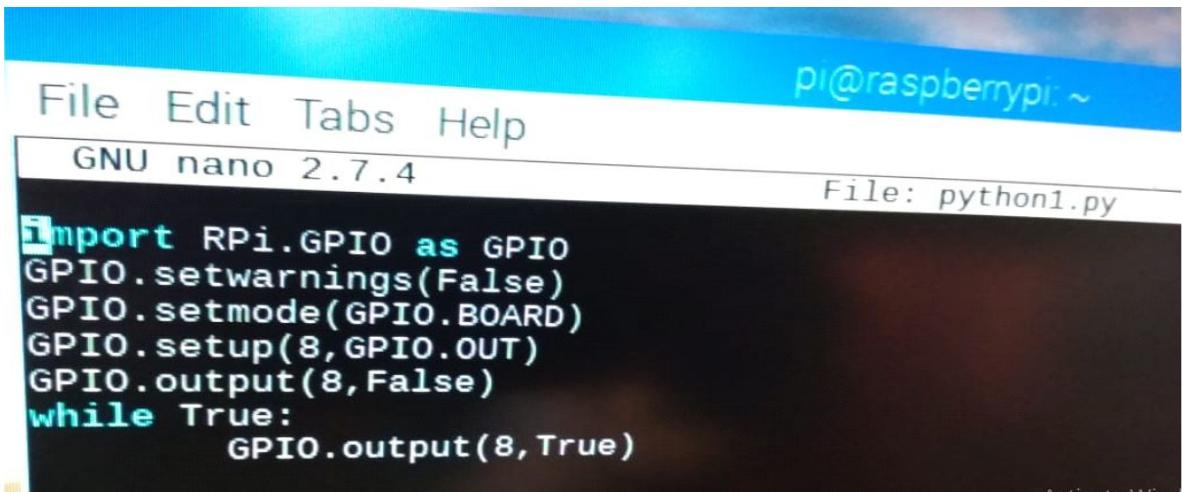
- Power on your RPi and boot all the way into the operating system GUI
- Open the terminal and launch the IDLE IDE as a superuser: `sudo idle`
- Wait for IDLE to open, then click **File > New** to open a new window (Ctrl + N)

- Type the code into the window
- Click **File > Save** when you are done (Ctrl + S).
- To run your program, click **Run > Run** (Ctrl + F5). You should see your led light up! We just told the RPi to supply voltage (+3.3v) to our circuit using GPIO pin 7.

If you are having trouble getting the led to light up, double-check your wiring, and make sure you have installed the GPIO Python library according to my [instructions](#).

### **1) Write a Program to glow the LED at pin number 8 in BOARD mode.**

- Type the command “ sudo nano filename.py” to open the editor where you can write your python code.
- The below given image is the code for glowing the LED on.



The screenshot shows a terminal window titled "pi@raspberrypi: ~". The window has a blue header bar with the title and a white menu bar below it. The menu bar contains "File", "Edit", "Tabs", and "Help". The main area of the terminal shows the version "GNU nano 2.7.4" and the file name "File: python1.py". The code displayed is:

```

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

```

After writing the code press “ctrl + 0” to save the code and “ctrl + x” to exit from the editor.

- Once, you are back on the terminal enter the command “sudo python filename.py” to run your program . If your connections are proper and the code is written correctly the LED will be glowing .
- The Output looks like this:



## 2) **Write a Program to glow the LED at pin number 8 in BCM mode.**

- **Program :**

```
File Edit Terms Help
GNU nano 2.7.4

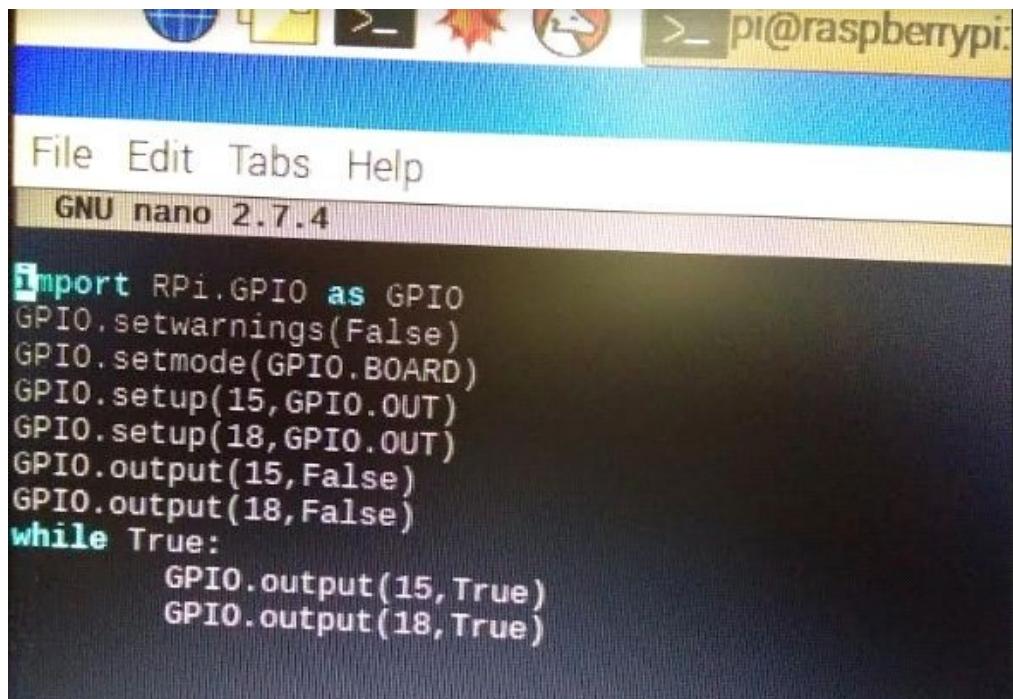
import RPi.GPIO as GPIO
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(14,GPIO.OUT)
GPIO.output(14,False)
while True:
    GPIO.output(14,True)
```

### Output :



### 3) Write a Program to glow any two LED's in BOARD mode.

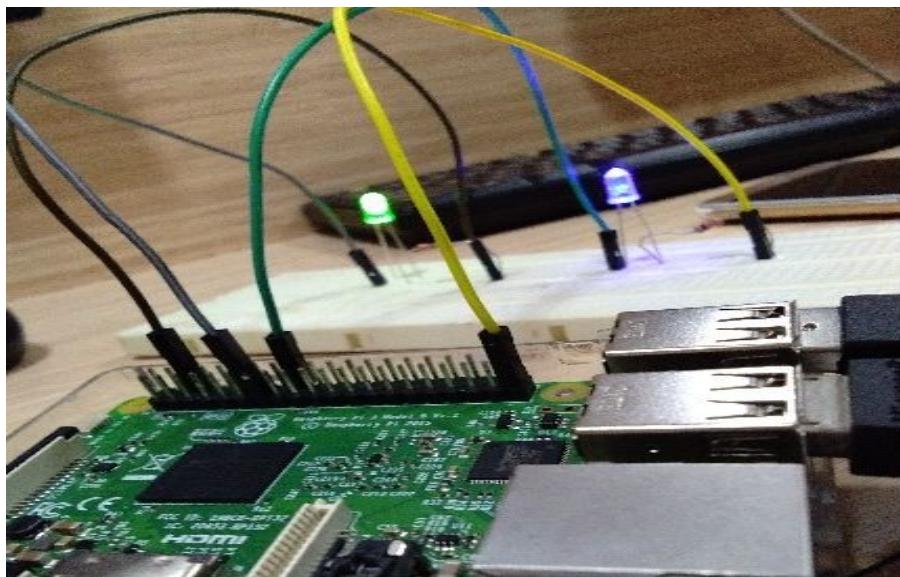
- **Program :**



The screenshot shows a terminal window titled 'pi@raspberrypi:'. The window has a blue header bar with icons for file operations and a yellow status bar. Below the header is a menu bar with 'File', 'Edit', 'Tabs', and 'Help'. The main area is titled 'GNU nano 2.7.4'. The code displayed is:

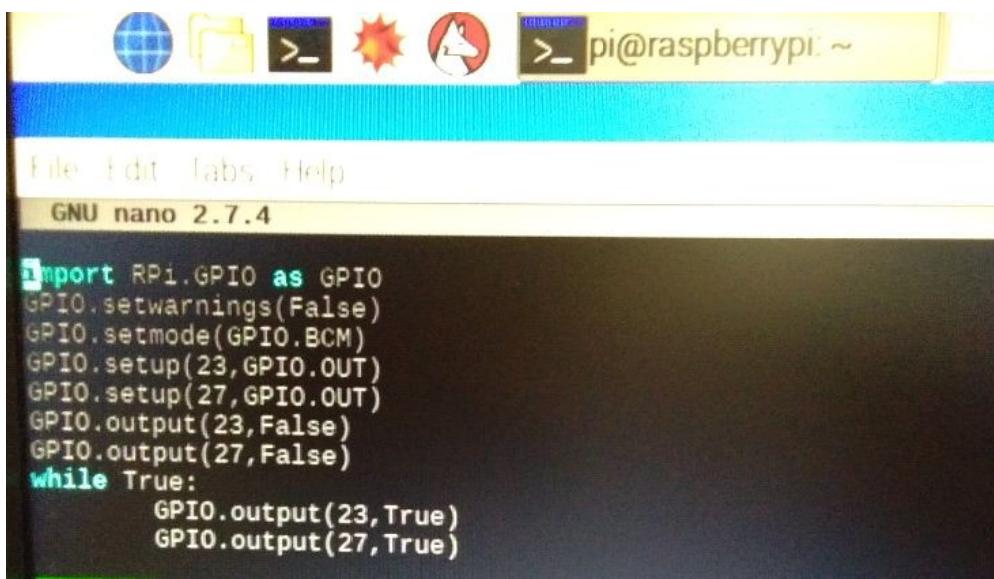
```
Import RPi.GPIO as GPIO
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(15,GPIO.OUT)
GPIO.setup(18,GPIO.OUT)
GPIO.output(15,False)
GPIO.output(18,False)
while True:
    GPIO.output(15,True)
    GPIO.output(18,True)
```

### Output :



**4) Write a Program to glow any two LED's in BCM mode.**

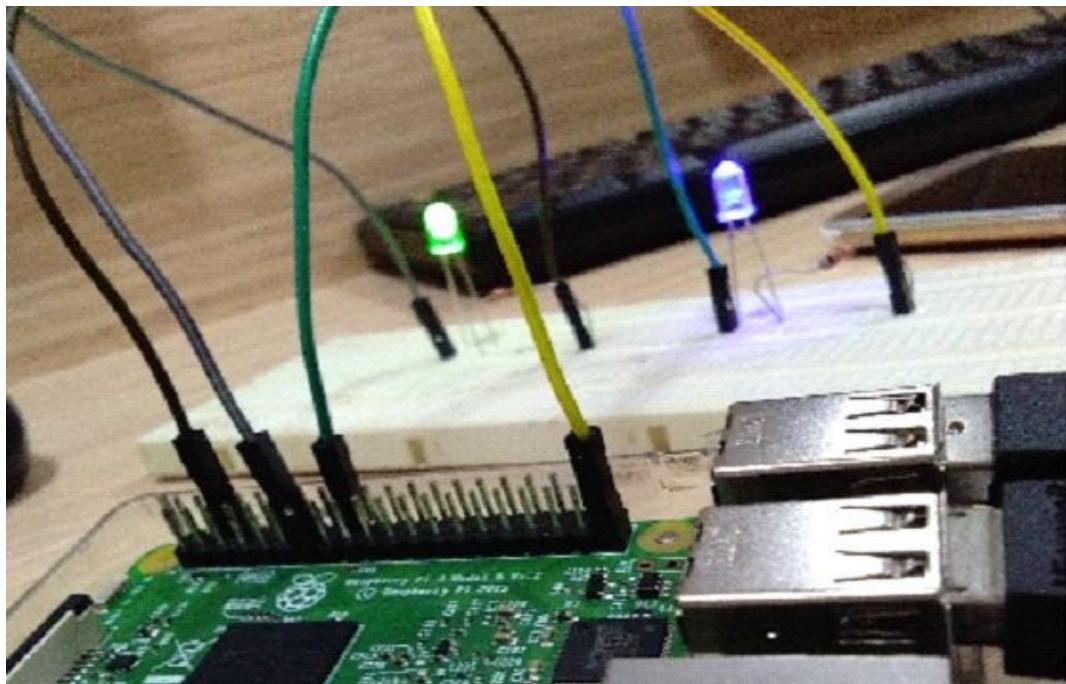
- **Program :**



The screenshot shows a terminal window titled "pi@raspberrypi: ~" with a blue header bar containing icons for network, file, terminal, and system status. Below the title is a menu bar with "File", "Edit", "Tabs", and "Help". The main area is titled "GNU nano 2.7.4". The code displayed is:

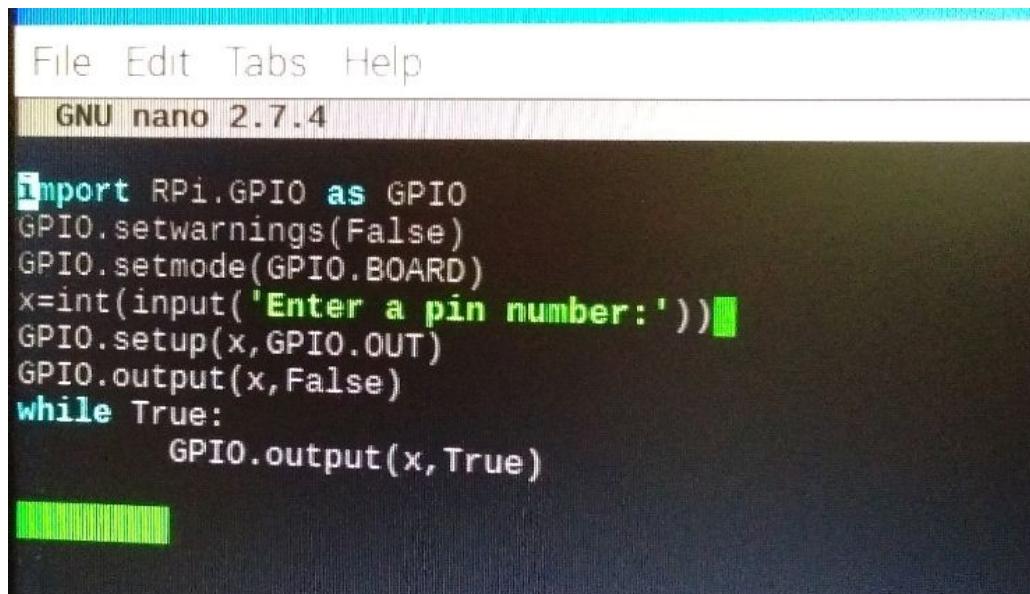
```
import RPi.GPIO as GPIO
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(23,GPIO.OUT)
GPIO.setup(27,GPIO.OUT)
GPIO.output(23,False)
GPIO.output(27,False)
while True:
    GPIO.output(23,True)
    GPIO.output(27,True)
```

**Output :**



**5) Write a Program to glow the LED by accepting pin number from user in BOARD mode.**

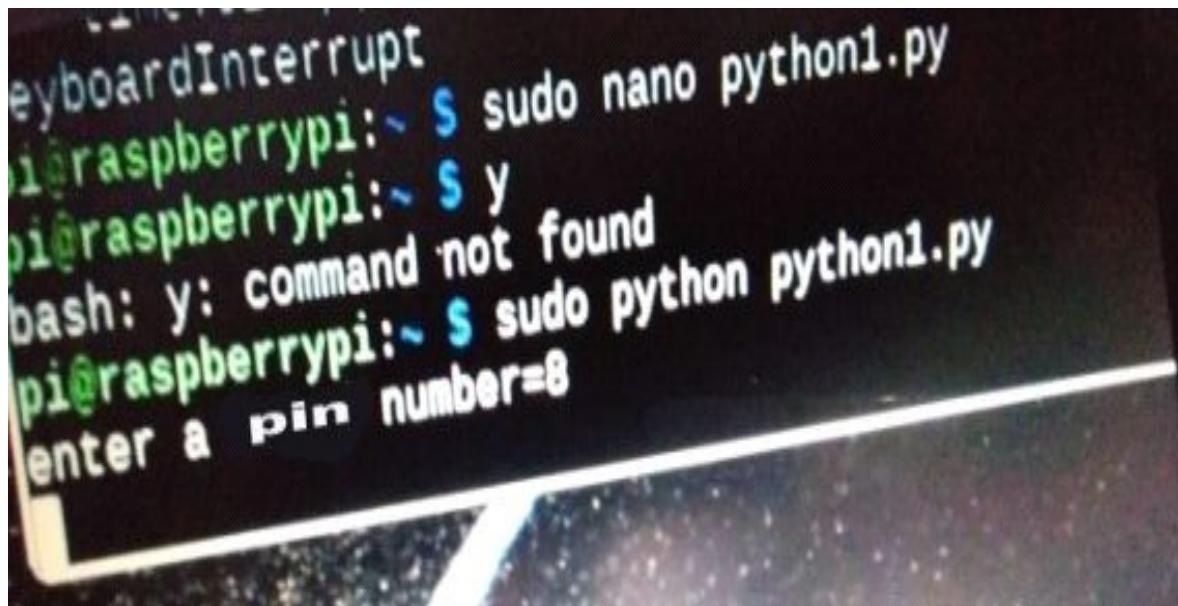
- **Program :**



The screenshot shows a terminal window titled "GNU nano 2.7.4". The script content is as follows:

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

**Output :**



The screenshot shows a terminal window with the following session:

```
pi@raspberrypi:~ $ sudo nano python1.py
pi@raspberrypi:~ $ y
pi@raspberrypi:~ $ bash: y: command not found
pi@raspberrypi:~ $ sudo python python1.py
pi@raspberrypi:~ $ Enter a pin number=8
```

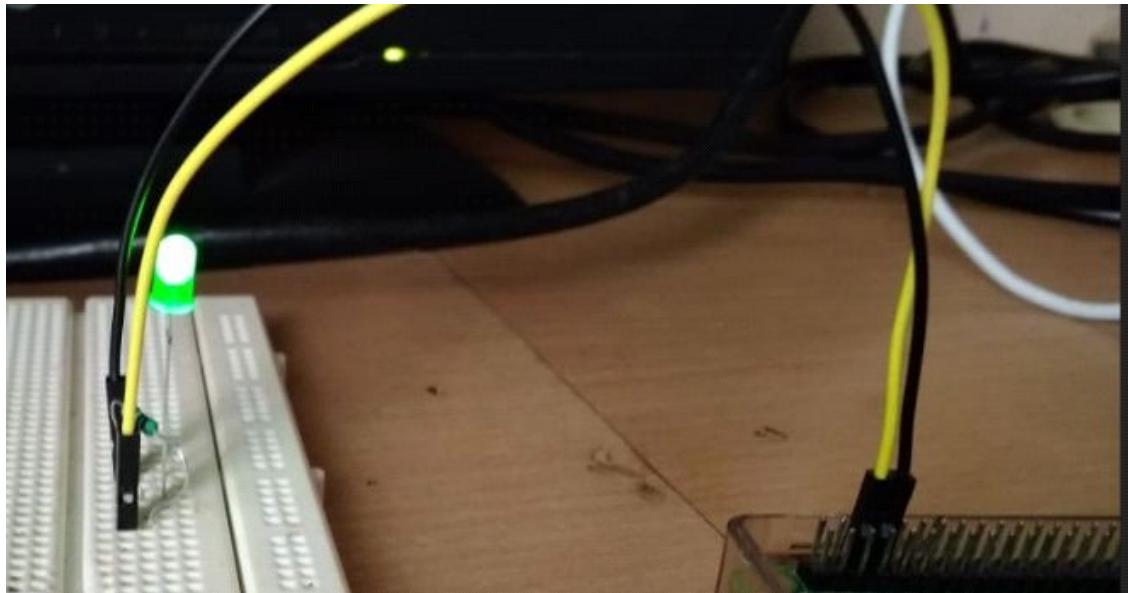


## 6) Write a Program to blink the LED at pin number 8 in BOARD mode.

- **Program :**

```
File: python1.py
GNU nano 2.7.4
import RPi.GPIO as GPIO
import time
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(8,GPIO.OUT)
GPIO.output(8,False)
while True:
    GPIO.output(8,False)
    time.sleep(1)
    GPIO.output(8,True)
    time.sleep(1)
```

## **Output :**



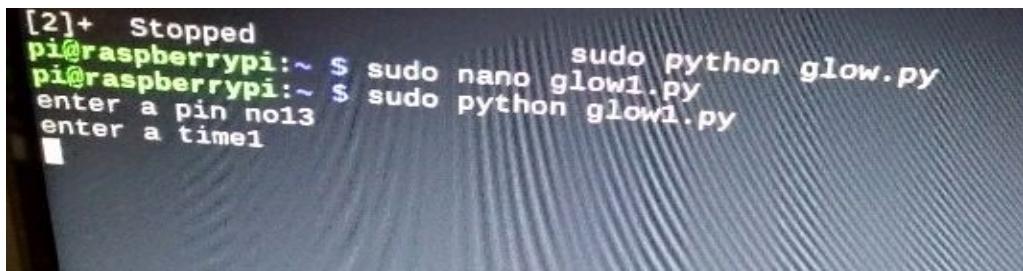
### **7) Write a Program to blink the LED accepted pin number and time from user.**

- Program :**

```
File Edit Tabs Help
GNU nano 2.7.4

import RPi.GPIO as GPIO
import time
a=int(input("enter a pin no"))
x=int(input("enter a time"))
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(a,GPIO.OUT)
while True:
    GPIO.output(a,False)
    time.sleep(x)
    GPIO.output(a,True)
    time.sleep(x)
```

## **Output :**



```
[2]+ Stopped sudo python glow.py
pi@raspberrypi:~ $ sudo nano glow1.py
pi@raspberrypi:~ $ sudo python glow1.py
enter a pin no13
enter a time1
```



### **8) Write a Program to glow the LED when you enter a even number .**

- **Program :**

```
GNU nano 2.7.4

import RPi.GPIO as GPIO
a=int(input("enter the pin no."))
if(a%2==0):
    print("led will glow")
    GPIO.setwarnings(False)
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(8,GPIO.OUT)
    GPIO.output(8,False)
while True:
    GPIO.output(8,True)
else:
    print("sorry led can't glow")
```

## Output :

```
[1]+ Stopped                 sudo python glow.py
pi@raspberrypi:~ $ sudo nano glow.py
pi@raspberrypi:~ $ sudo python glow.py
enter the pin no.4
led will glow
```



## PRACTICAL NO. 4

### **GPIO: LED Grid Module: Program the 8X8 Grid with different formulas .**

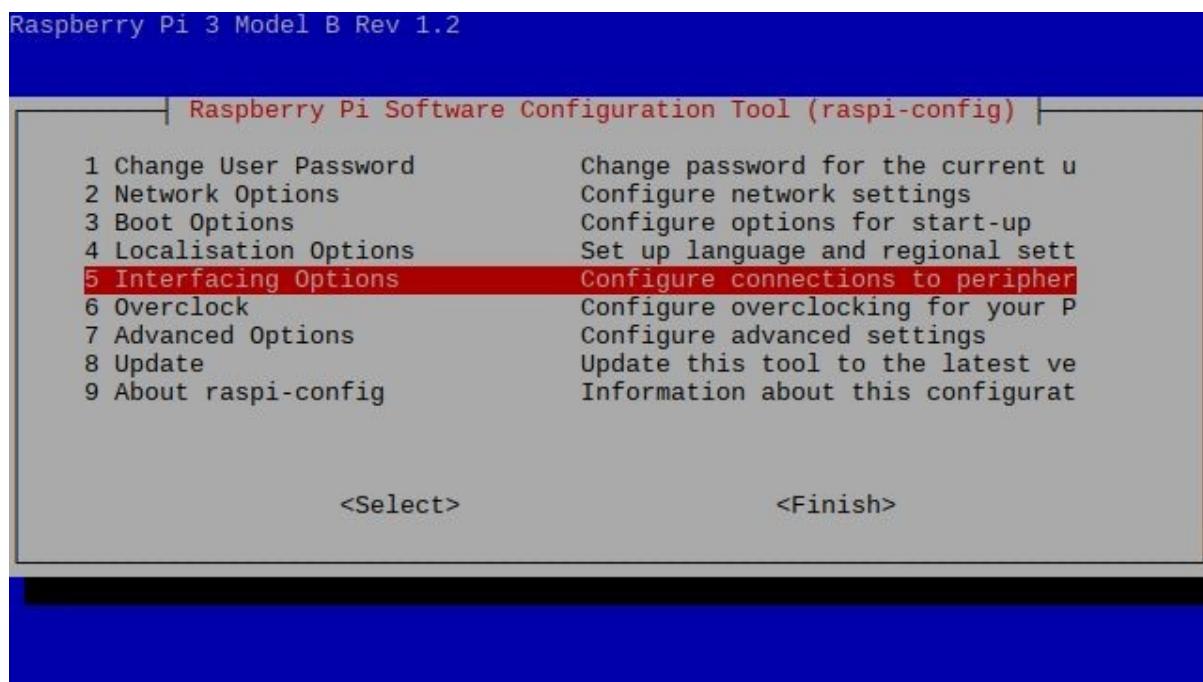
The 8x8 LED Matrix and the max7219 chip allow us to control 64 LEDs per module. Because of the max7219 chip on the module we don't need to connect every single led to the Raspberry Pi. We only need to connect the chip to the Pi and the chip will control all of the LEDs on our behalf. The SPI communication protocol will be utilized to provide communication between the chip and the Pi, following are the steps to activate SPI

Step 1:

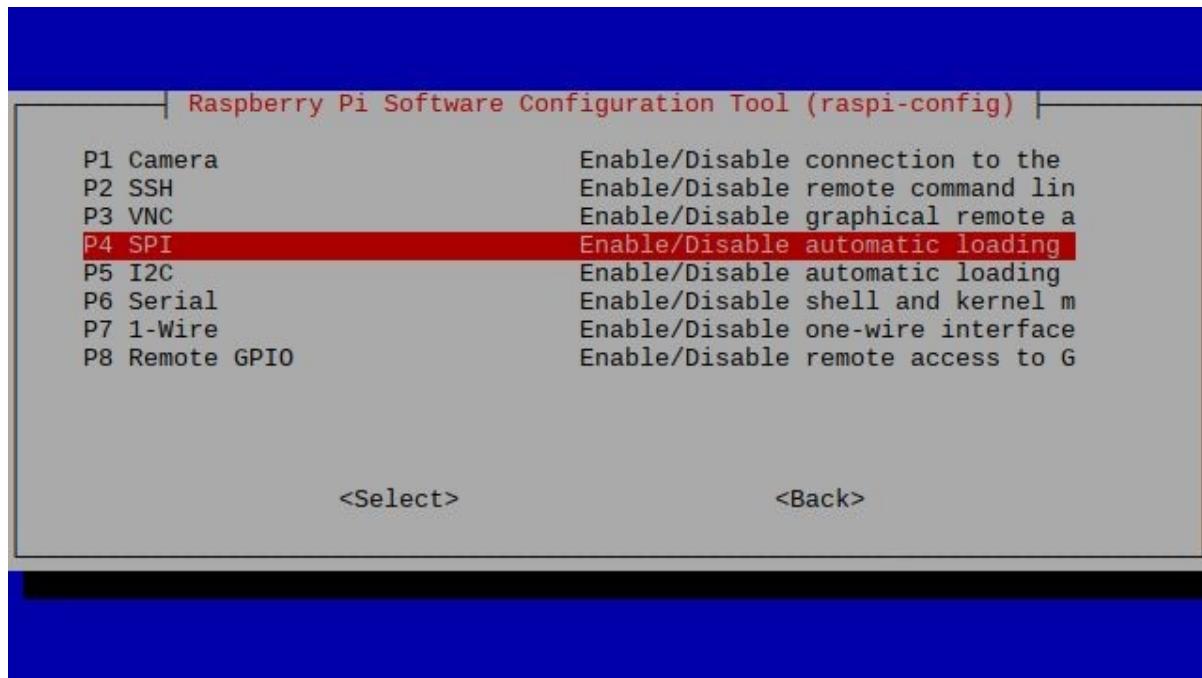
From the command line or Terminal window start by running the following command :

Sudo raspi-config

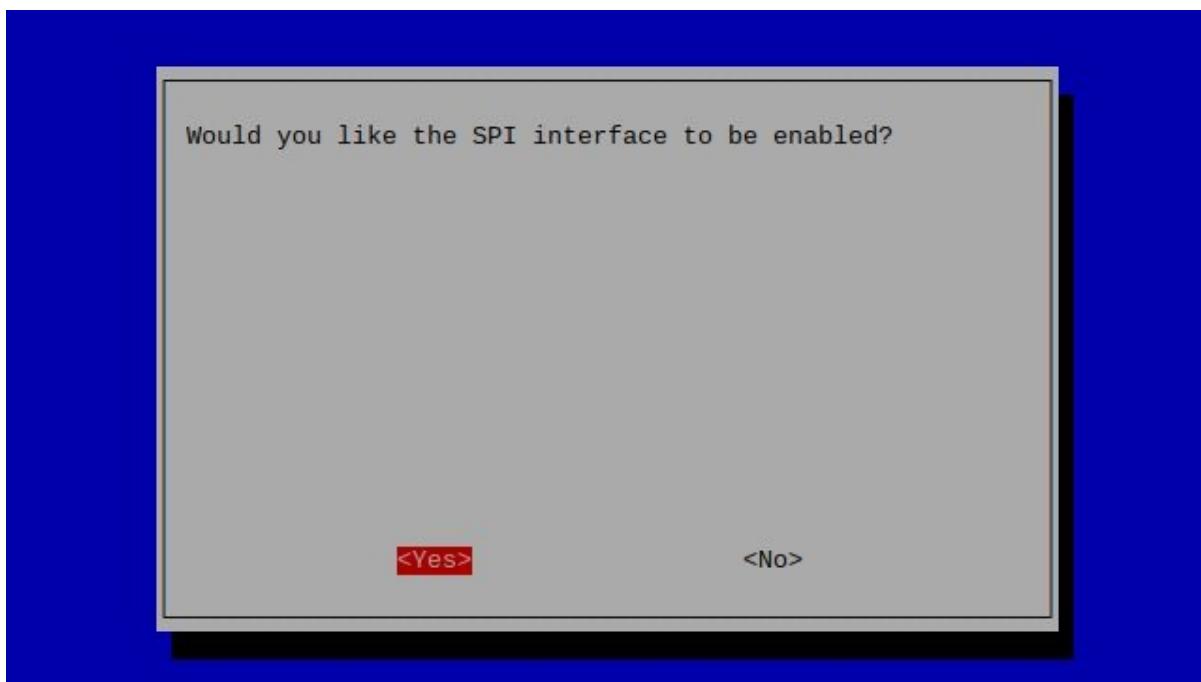
This will launch the raspi-config utility. Select "Interfacing Options" :



Highlight the "SPI" option and activate "<Select>".



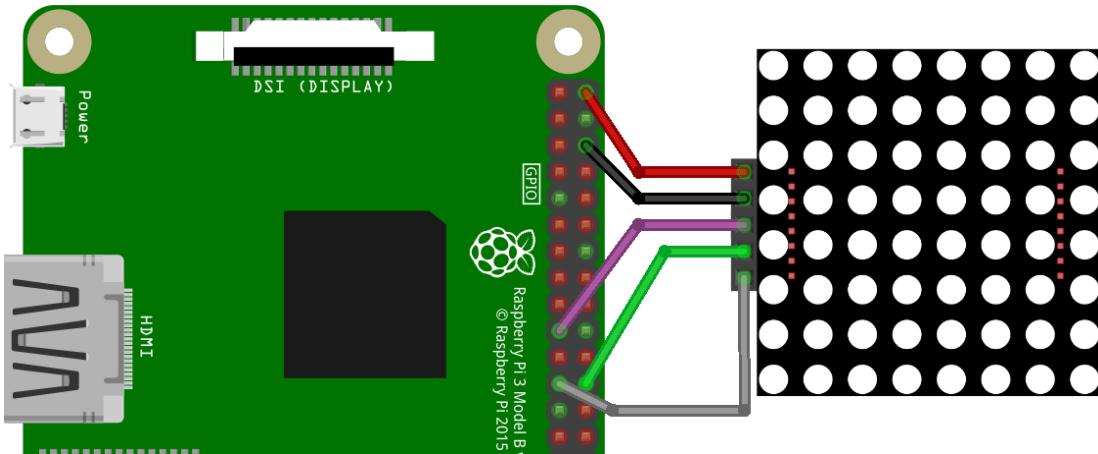
Select and activate "<Yes>" :



Now click on finish and your spi is enabled .

## Step 2 : Connections

Now that we have enabled the spi, we next need to connect the 8x8 LED matrix to the Raspberry Pi.



**Connections Raspberry Pi**

5V

GND

GPIO 10 (MOSI)

GPIO 8 (SPI CE0)

GPIO 11 (SPI CLK)

**Connections LED Matrix**

1 - VCC

2 - GND

3 - DIN

4 - CS

5 - CLK

Step 3: After connections, we have to install drivers, below given are the following commands .

```
git clone https://github.com/rm-hull/max7219.git sudo python max7219/setup.py install
```

```
pi@raspberrypi: ~ git clone https://github.com/rm-hull/max7219.git
Cloning into 'max7219'...
remote: Enumerating objects: 1, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 1884 (delta 0), reused 0 (delta 0), pack-reused 1883
Receiving objects: 100% (1884/1884), 11.25 MiB | 617.00 KiB/s, done.
Resolving deltas: 100% (1096/1096), done.
pi@raspberrypi: ~ cd max7219
pi@raspberrypi:~/max7219 $ sudo python setup.py install

running install
running bdist_egg
running egg_info
creating luma.led_matrix.egg-info
writing requirements to luma.led_matrix.egg-info/requirements.txt
writing luma.led_matrix.egg-info/PKG-INFO
writing top-level names to luma.led_matrix.egg-info/top_level.txt
writing dependency_links to luma.led_matrix.egg-info/dependency_links.txt
writing manifest file 'luma.led_matrix.egg-info/SOURCES.txt'
reading manifest file 'luma.led_matrix.egg-info/SOURCES.txt'
reading manifest template 'MANIFEST.in'
no previously-included directories found matching 'doc/build'
warning: no previously-included files matching '__pycache__' found under directory ''
warning: no previously-included files matching '__pycache__.pyc' found under directory ''
warning: no previously-included files matching '__pycache__.pyo' found under directory ''
warning: no previously-included files matching '__pycache__.pyd' found under directory ''
warning: no previously-included files matching '.DS_Store' found under directory ''
warning: no previously-included files matching '.ropeproject' found under directory ''
writing manifest file 'luma.led_matrix.egg-info/SOURCES.txt'
installing library code to build/bdist.linux-armv7l/egg
running install_lib
running build_py
creating build
creating build/lib.linux-armv7l-2.7
creating build/lib.linux-armv7l-2.7/luma
copying luma/_init_.py -> build/lib.linux-armv7l-2.7/luma
creating build/lib.linux-armv7l-2.7/luma/led_matrix
```

#### Step 4: Executing the program

As the drivers are installed, run the demo program in the examples subdirectory which is in max7219 directory , following is the cmd to enter into the examples directory.

```
cd max7219/examples
```

After entering into examples folder, execute matrix.py Sudo python matrix.py

```
[ 2202.711659] Under-voltage detected! (0x00050005)
[ 2208.951596] Under-voltage detected! (0x00050005)/m...
[ 2217.271578] Under-voltage detected! (0x00050005)
pi@raspberrypi: ~/max7219/examples
File Edit Tabs Help
pi@raspberrypi:~/max7219 $ cd examples
pi@raspberrypi:~/max7219/examples $ python matrix_demo.py
Created device
MAX7219 LED Matrix Demo
Fast scrolling: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, qui
s nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat n
ulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
Slow scrolling: The quick brown fox jumps over the lazy dog
```

## PRACTICAL NO.5

### SPI: Camera Connection and capturing Images using SPI



Step 1 :

Camera slot port is available in between audio port and HDMI port insert the camera in the slot.

Step 2 :

Just pull the camera and check the connection.

Step 3 :

Update & upgrade the raspberry pi by using following commands.

“sudo apt-get update”

“sudo apt-get upgrade”

Step 4 :

Run: “sudo raspi-config”

Go to the camera option & enable it.

Step 5 :

To take a still picture follow the below given commands. “raspistill -o image.jpg”

Step 6 :

To capture the video follow the below given commands. “raspivid -o video.h264-t 1000”

To open the video.

“omxplayer video.h264”.

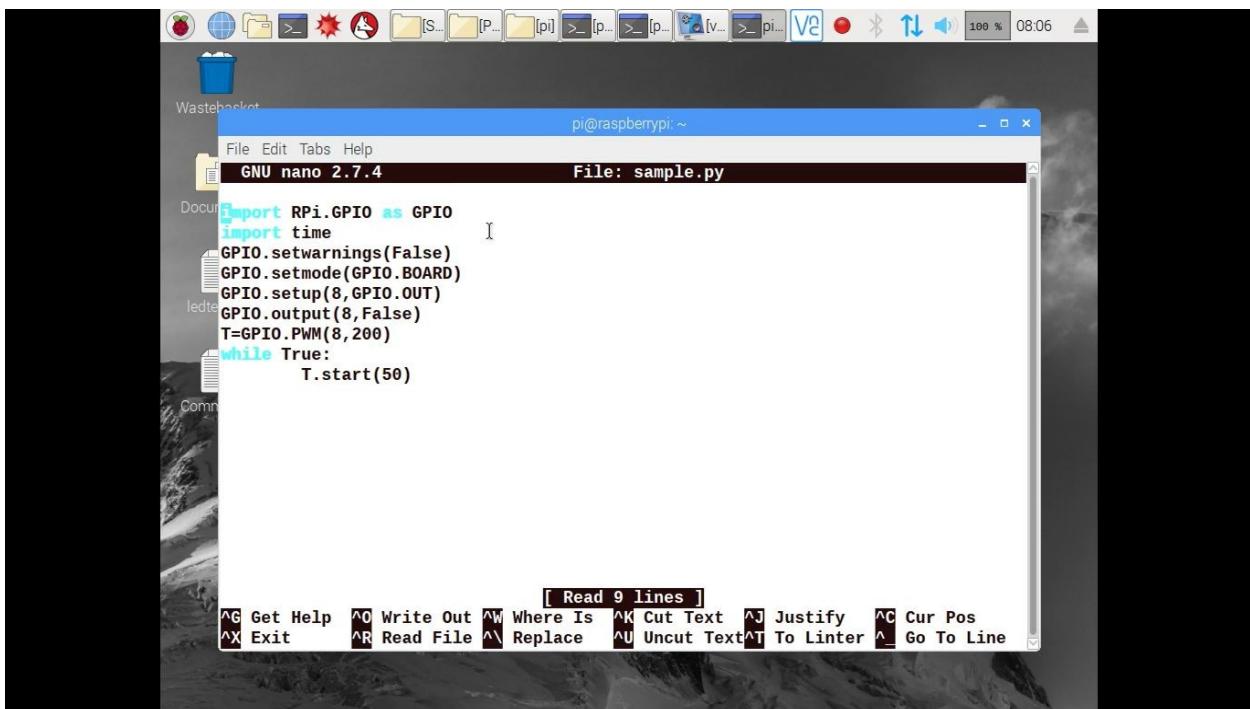
## PRACTICAL NO.6

### DEMONSTRATE THE USE OF PWM

- 1) WRITE A PROGRAM TO INCREASE THE INTENSITY OF AN LED BY USING PWM .

→

- Type the command “sudo nano filename.py” to open the editor where you can write your python code
- The below given image is the code for glowing the LED on.

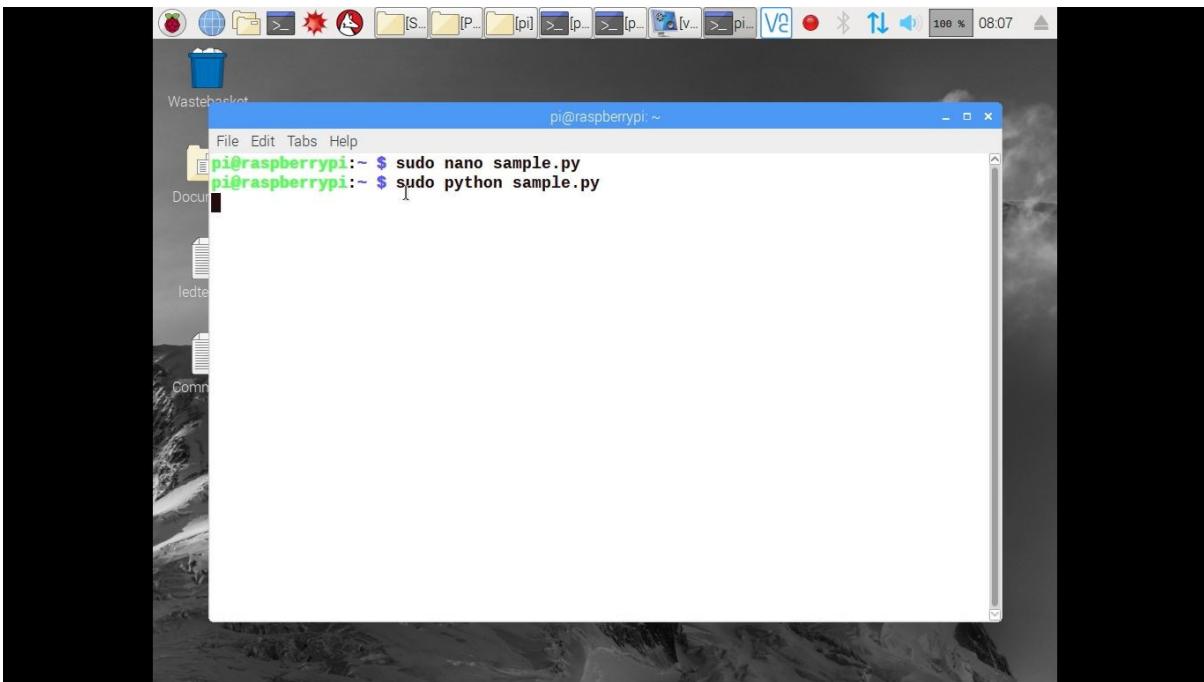


The screenshot shows a terminal window titled "GNU nano 2.7.4" with the file name "sample.py" open. The code in the window is:

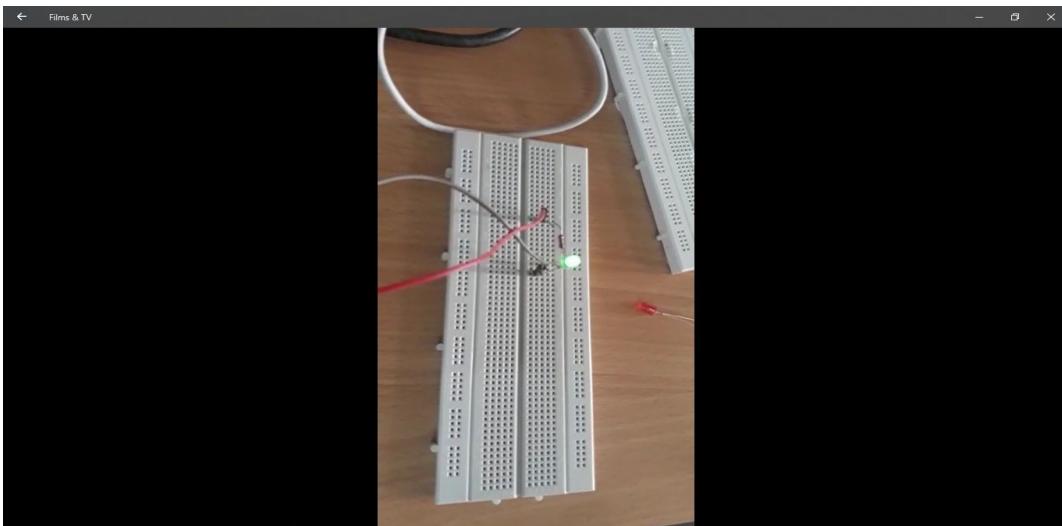
```
pi@raspberrypi: ~
File Edit Tabs Help
GNU nano 2.7.4 File: sample.py
Document  Import RPi.GPIO as GPIO
import time
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(8,GPIO.OUT)
GPIO.output(8,False)
T=GPIO.PWM(8,200)
while True:
    T.start(50)
```

The terminal window is part of a desktop environment on a Raspberry Pi, with various icons visible in the background.

- After writing the code press “ctrl+0” to save the code and “ctrl+X” to exit from the editor.
- Once you are back on the terminal enter the command “sudo python filename.py”



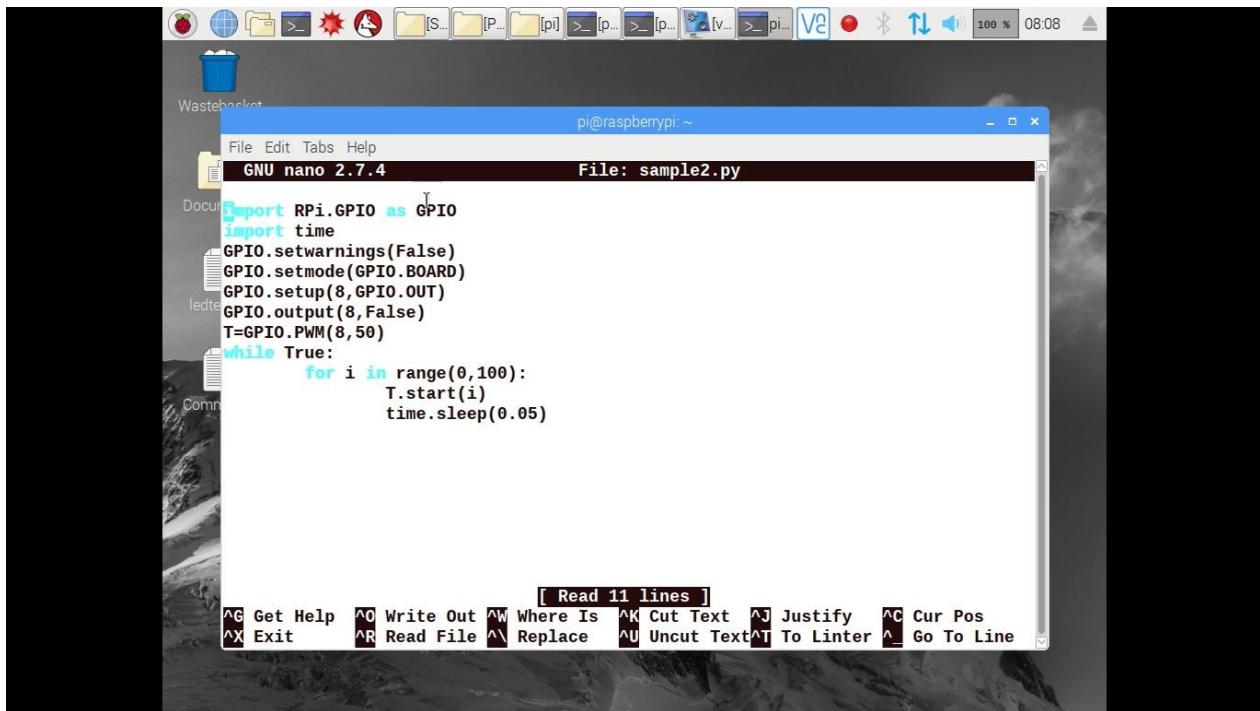
- to run your program.if your connections are proper and the code is written correctly the LED will be glowing.
- The output looks like this :



## 2) WRITE A PROGRAM TO INCREASE THE INTENCITY OF AN LED BY USING FOR LOOP IN PWM .

→

- Type the command “sudo nano filename.py” to open the editor where you can write your python code
- The below given image is the code for glowing the LED on :

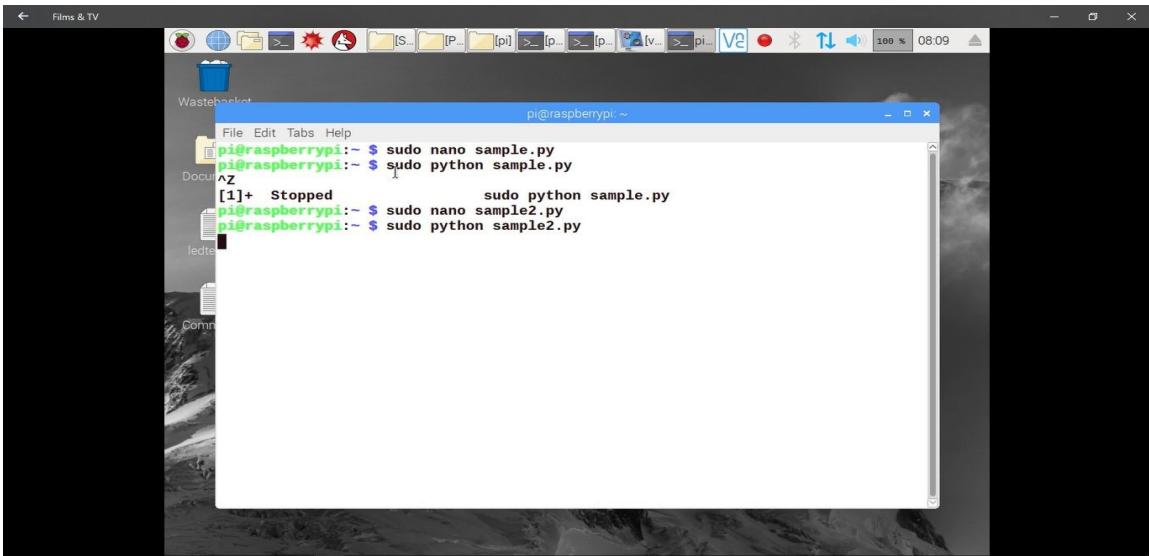


The screenshot shows a desktop environment on a Raspberry Pi. In the top bar, there are icons for a terminal, file browser, and various system services. The main window is a terminal window titled "pi@raspberrypi: ~". Inside the terminal, the "GNU nano 2.7.4" editor is open, displaying the following Python code:

```
import RPi.GPIO as GPIO
import time
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(8,GPIO.OUT)
GPIO.output(8,False)
T=GPIO.PWM(8,50)
while True:
    for i in range(0,100):
        T.start(i)
        time.sleep(0.05)
```

At the bottom of the terminal window, there is a menu bar with options like "File", "Edit", "Tabs", and "Help". Below the menu, there is a toolbar with icons for "Get Help", "Write Out", "Where Is", "Cut Text", "Justify", "Cur Pos", "Exit", "Read File", "Replace", "Uncut Text", "To Linter", and "Go To Line".

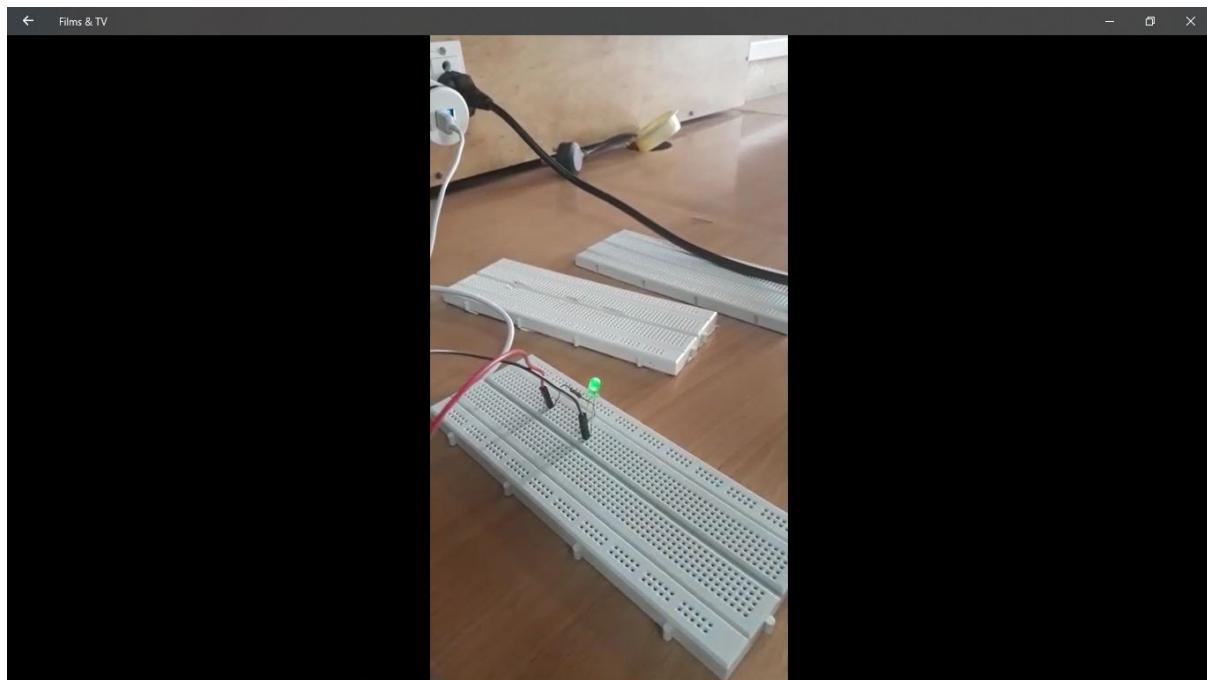
- After writing the code press “ctrl+0” to save the code and “ctrl+X” to exit from the editor.
- Once you are back on the terminal enter the command “sudo python filename.py”



A screenshot of a terminal window titled "Wastebasket" on a Raspberry Pi desktop environment. The window shows the following command-line session:

```
pi@raspberrypi:~ $ sudo nano sample.py
pi@raspberrypi:~ $ sudo python sample.py
[1]+  Stopped                  sudo python sample.py
pi@raspberrypi:~ $ sudo nano sample2.py
pi@raspberrypi:~ $ sudo python sample2.py
```

- to run your program.if your connections are proper and the code is written correctly the LED will be glowing.
- The output looks like this:



## PRACTICAL NO.7

### Node RED: Connect LED to Internet of Things

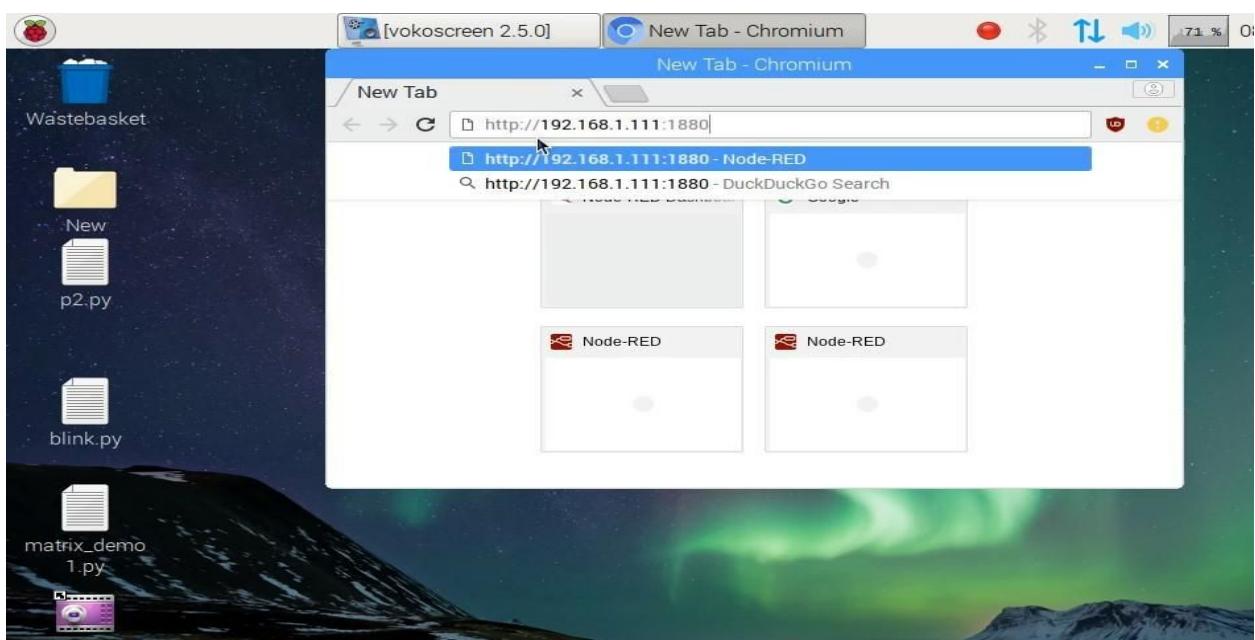
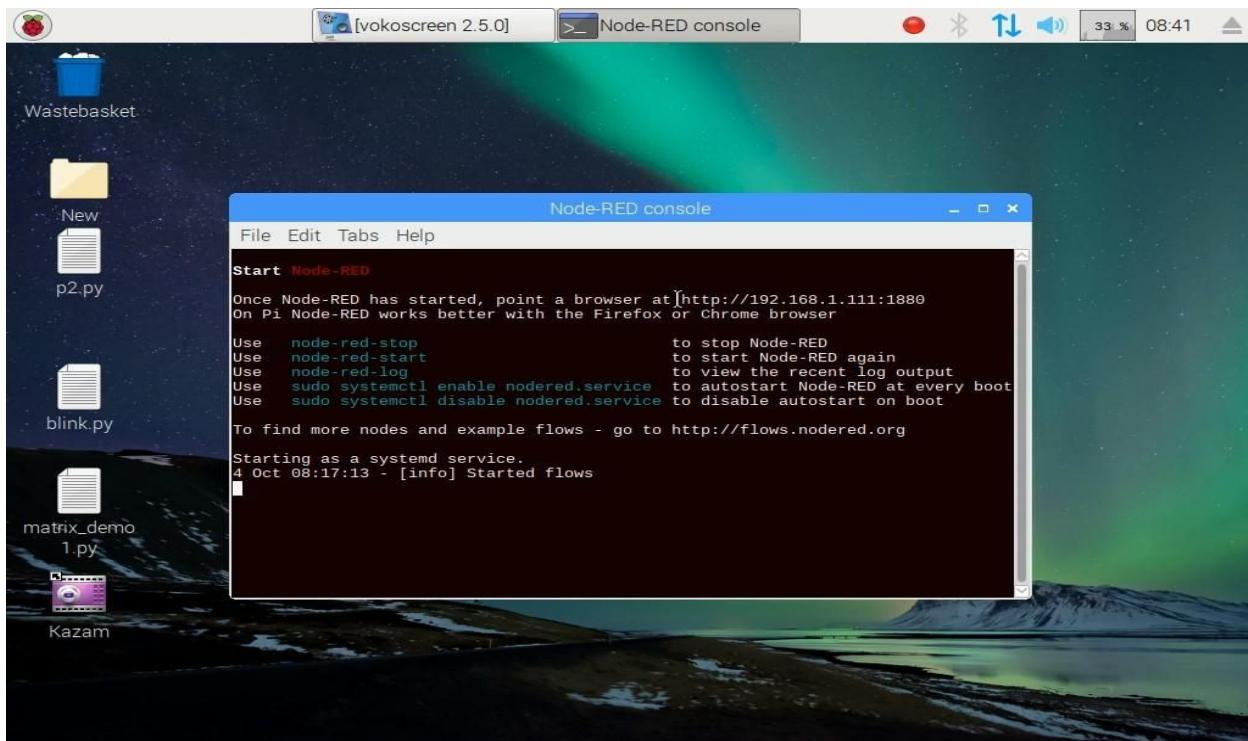
**Step 1 →**

**Go to Programming → Node-RED**



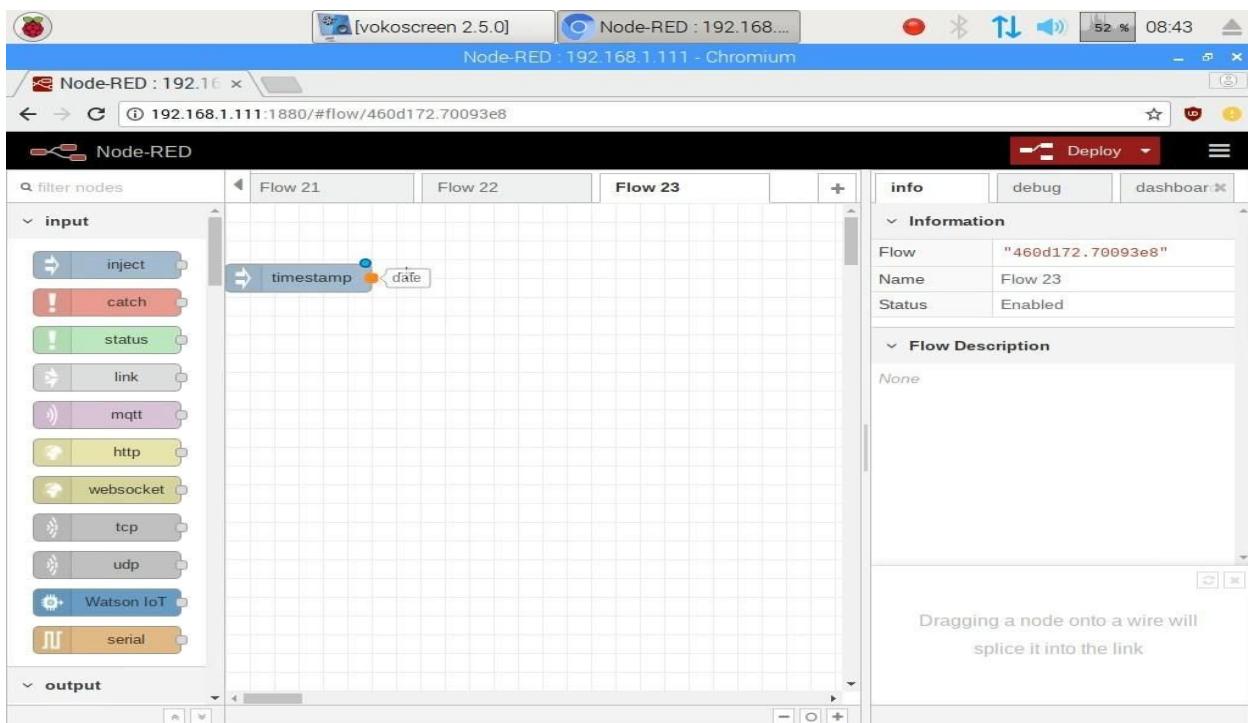
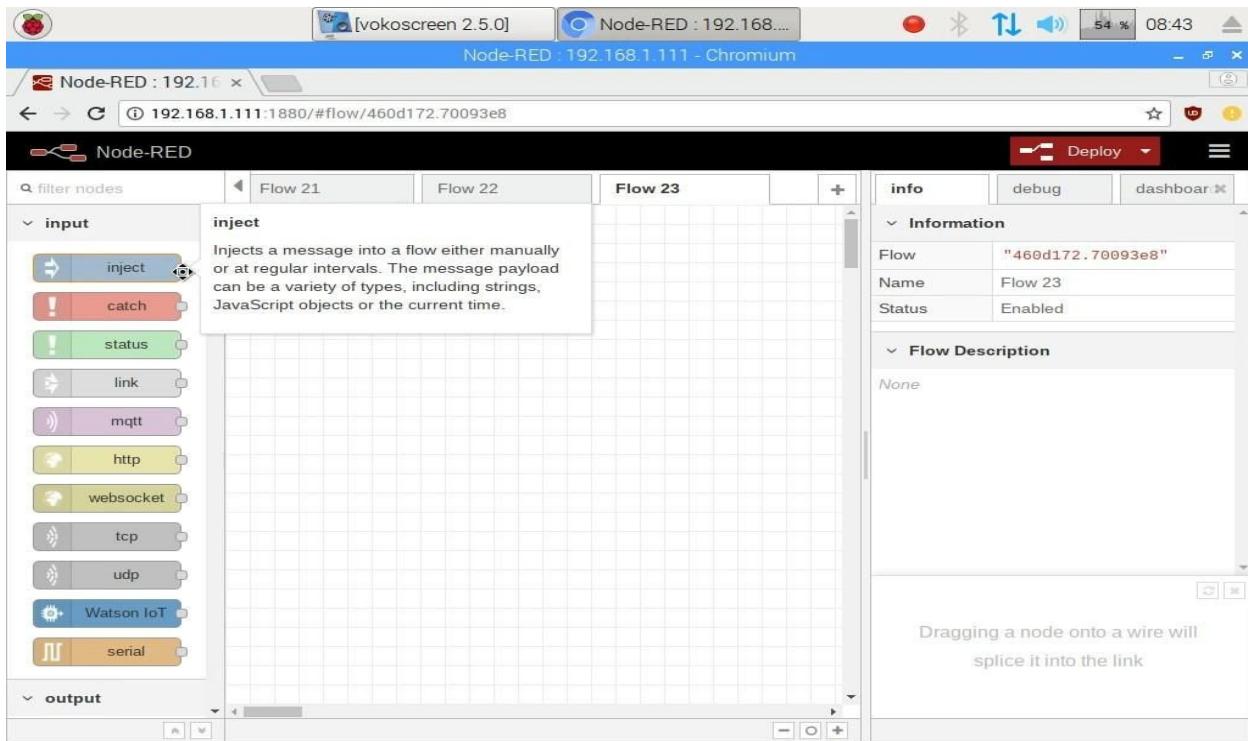
**Step 2 →**

**URL will display just copy the URL and past it at browser address bar then press enter.  
(<http://192.168.1.111:1880>).**



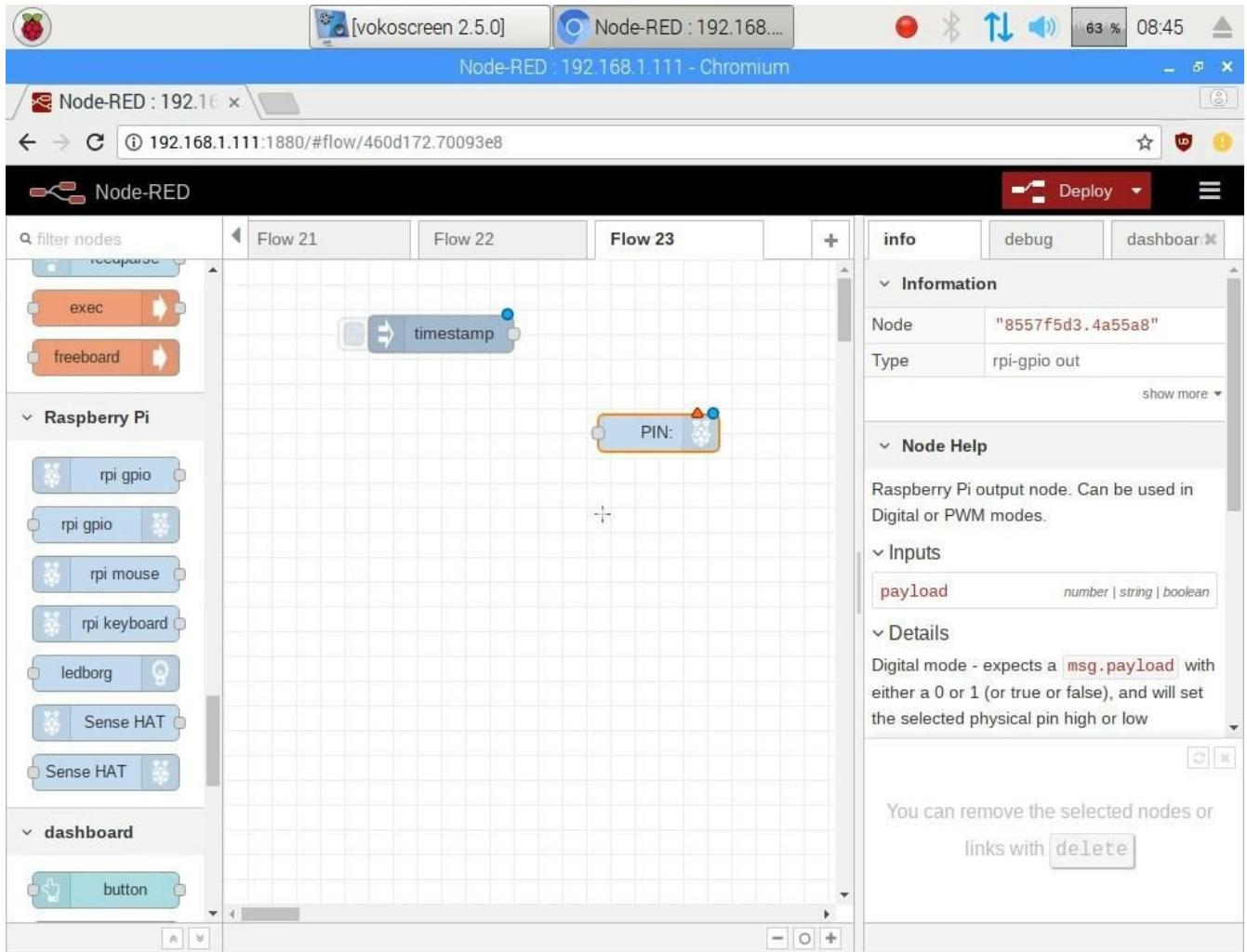
**Step 3→**

**Select Inject from Left side input bar.**



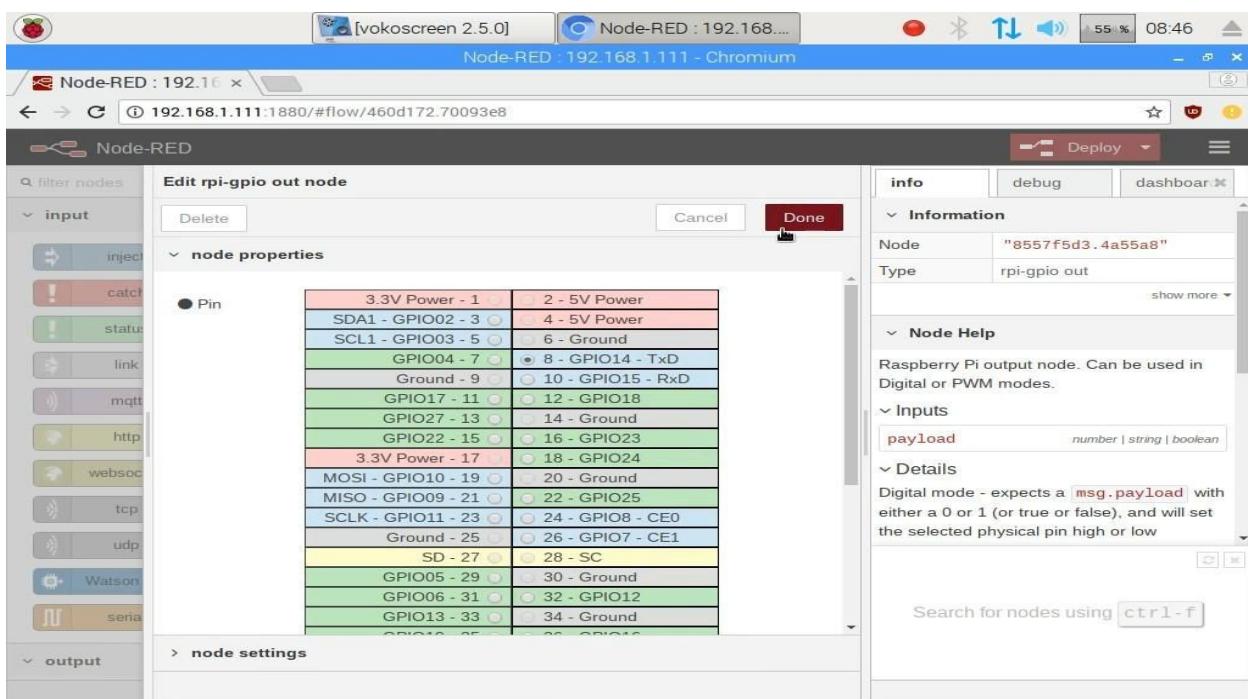
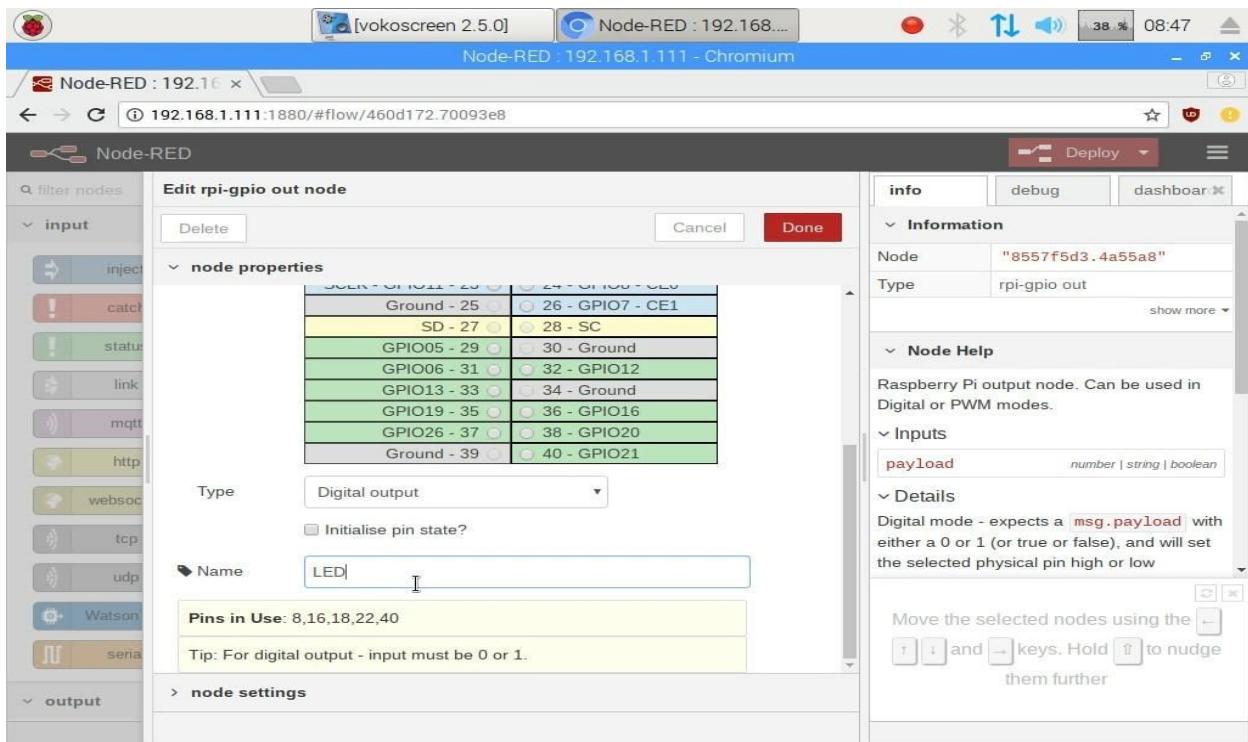
**Step 4 →**

**Go to Raspberry Pi bar which is available at the left side and select RP i.GPIO(Output).**



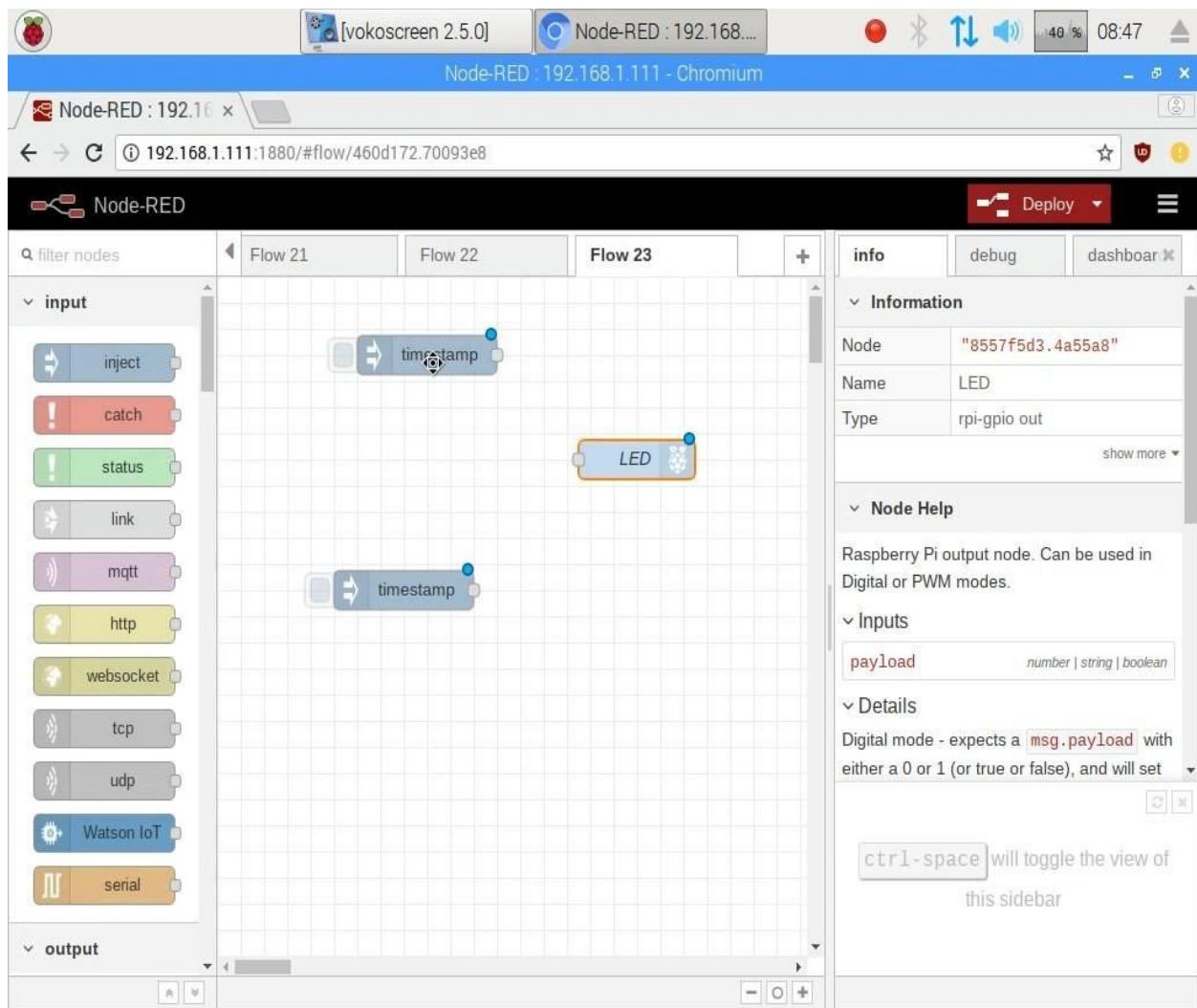
**Step 5 →**

**Double Click on RPi.GPIO and name it as LED and select the GPIO Pin no.8 .Type = Digital Output| |Click on Done.**



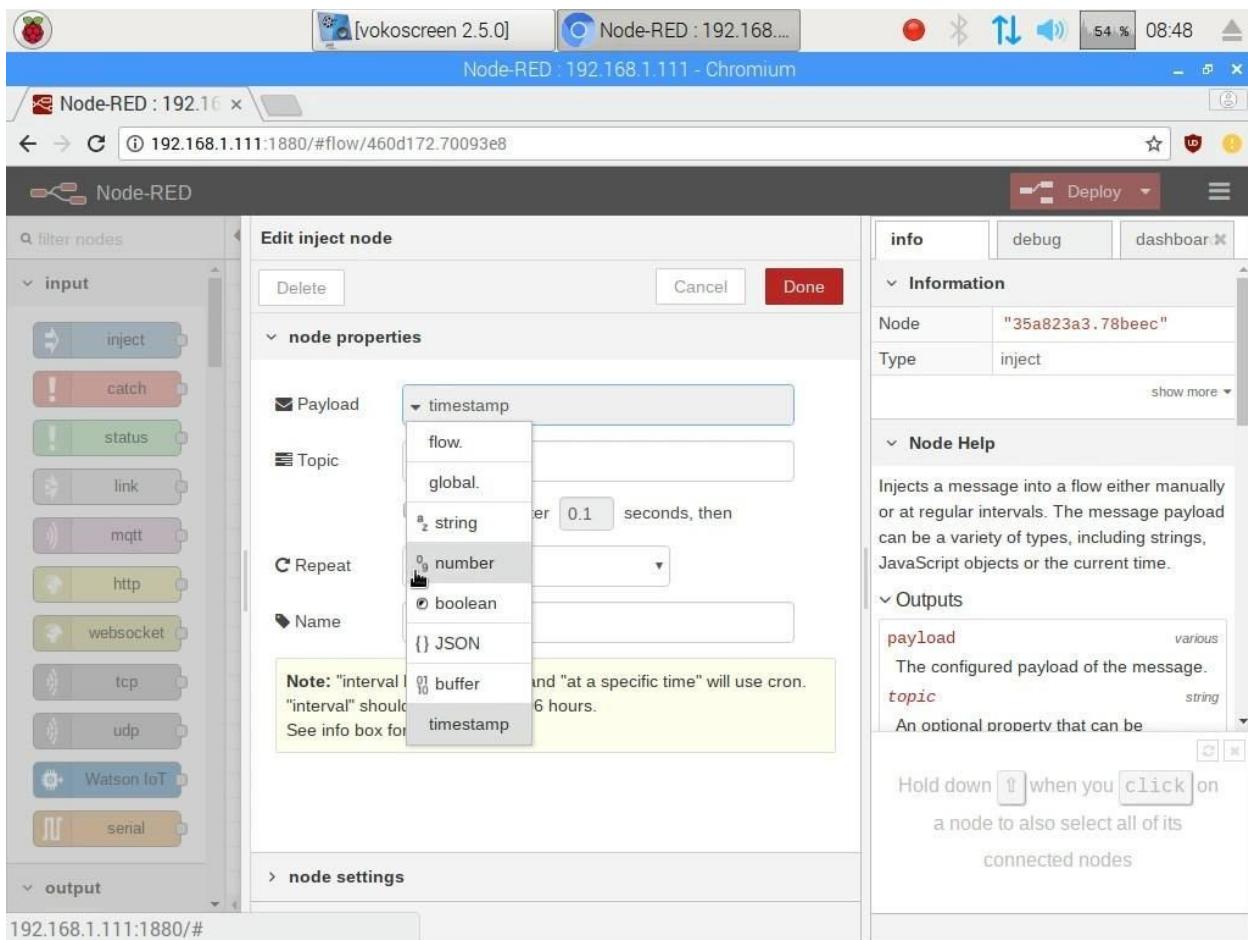
**Step 6 →**

**Take one more Inject button on the screen.**



**Step 7 →**

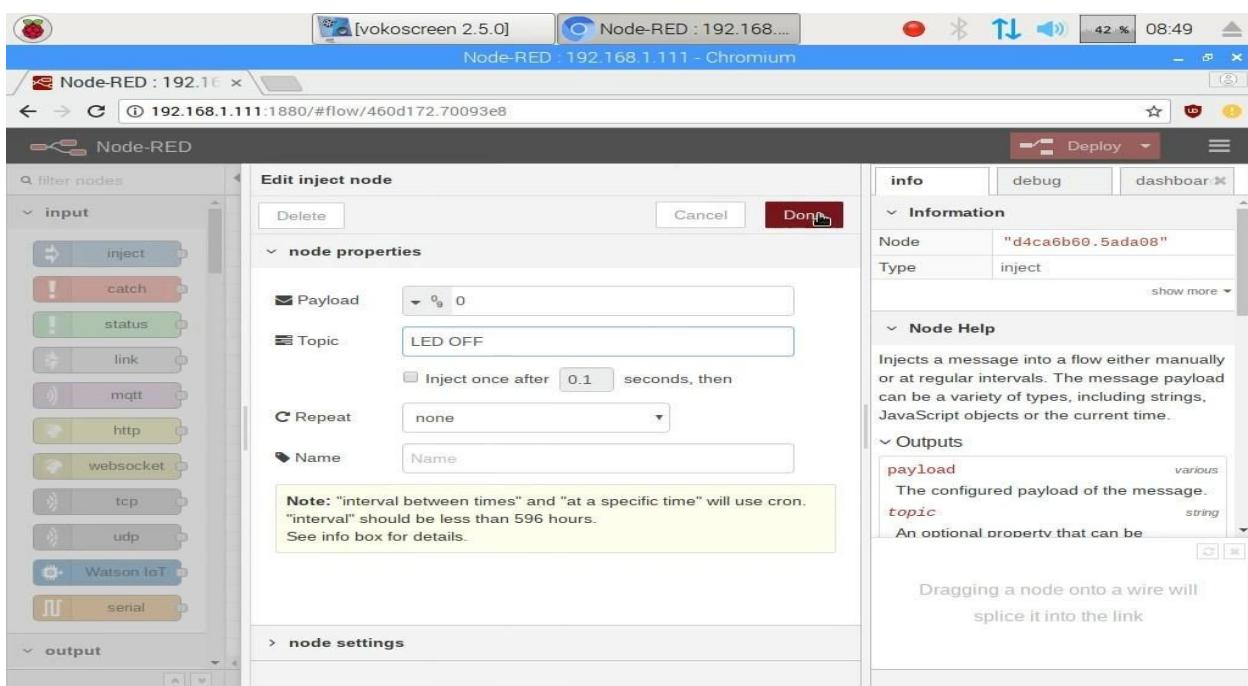
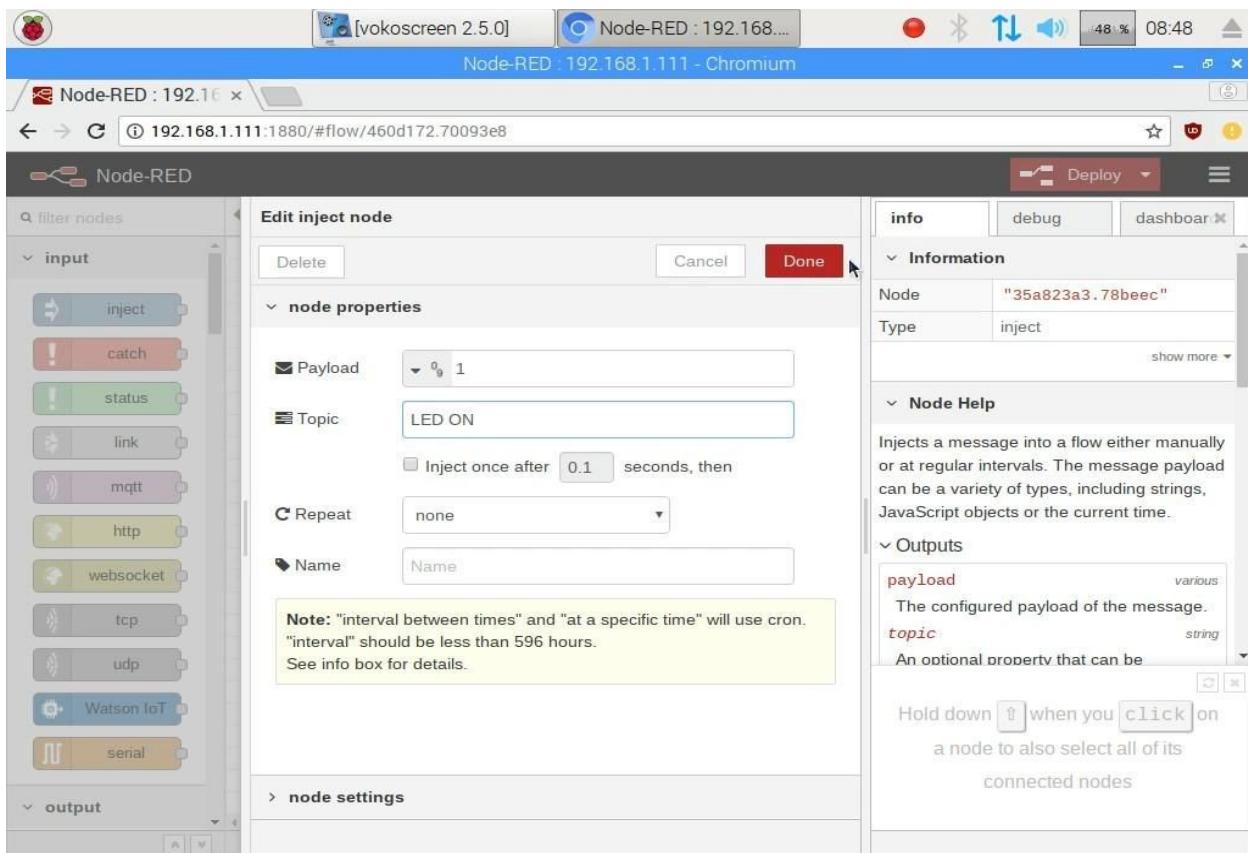
**Now double click on the inject button and select time number.**

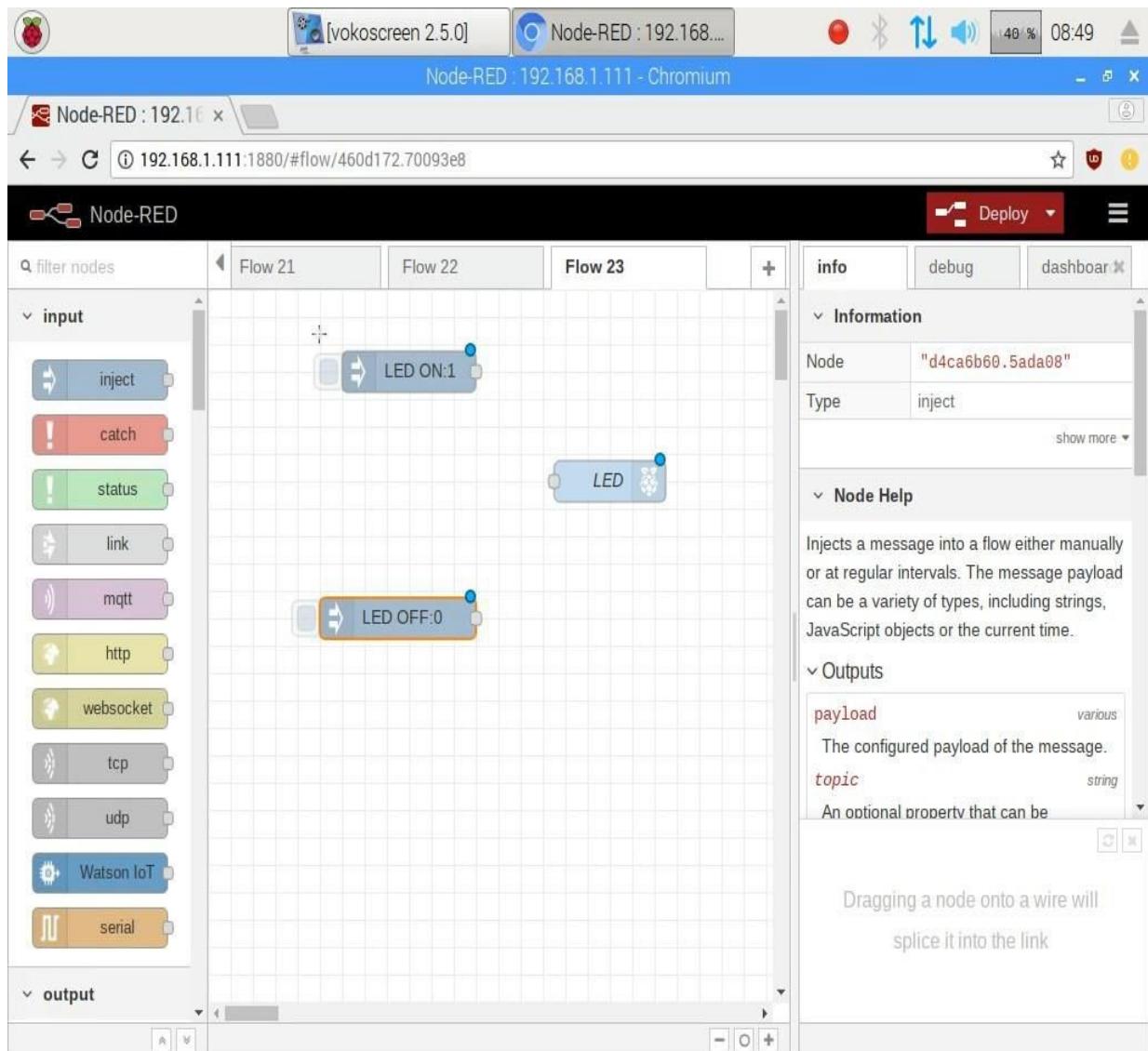


**Step 8 →**

**Name Inject Button as  
the following (Button1 =  
1 as ON Button 2 = 0 as  
OFF )**

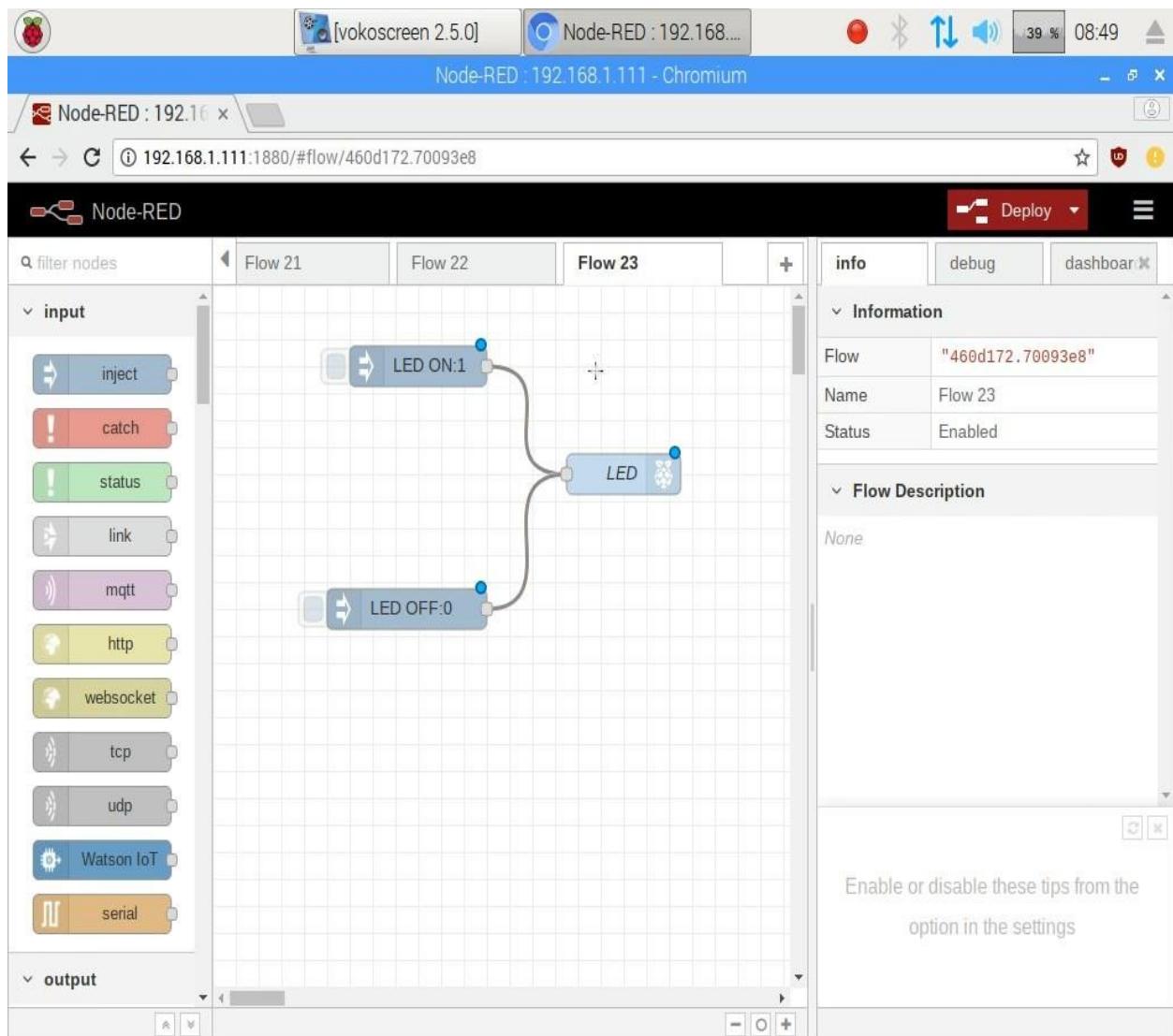
**Write 1 for ON in payload and 0 for OFF payload.**





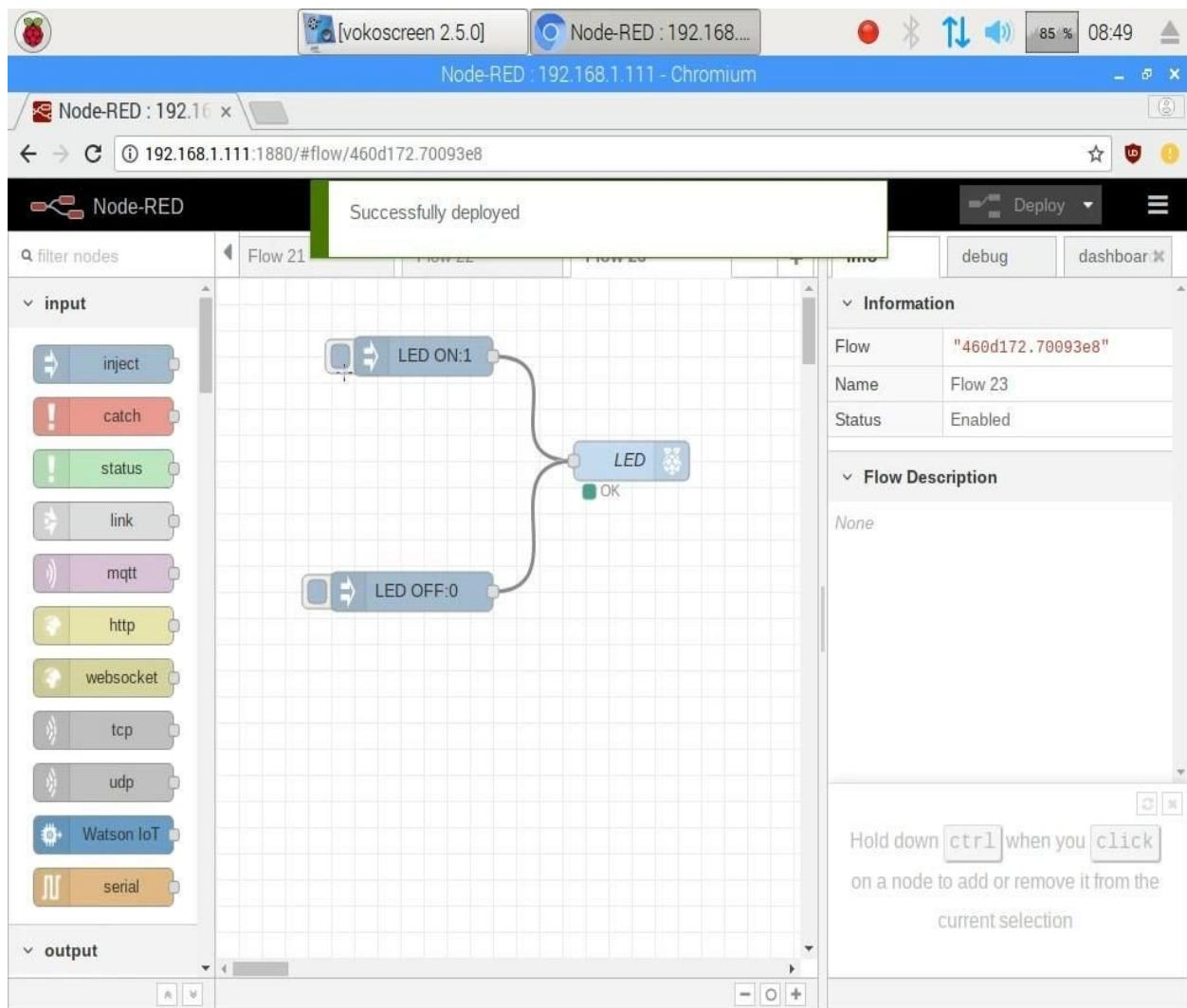
**Step 9 →**

**Connect first and second inject button to the LED.**



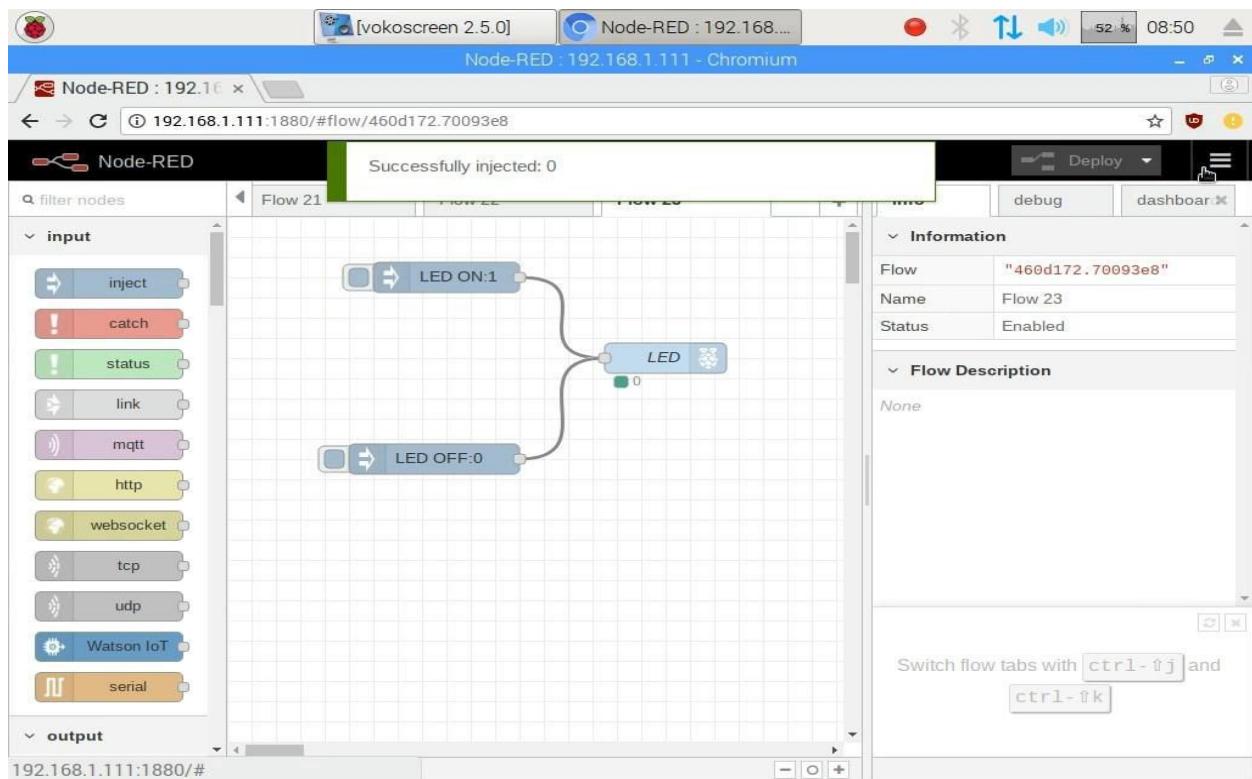
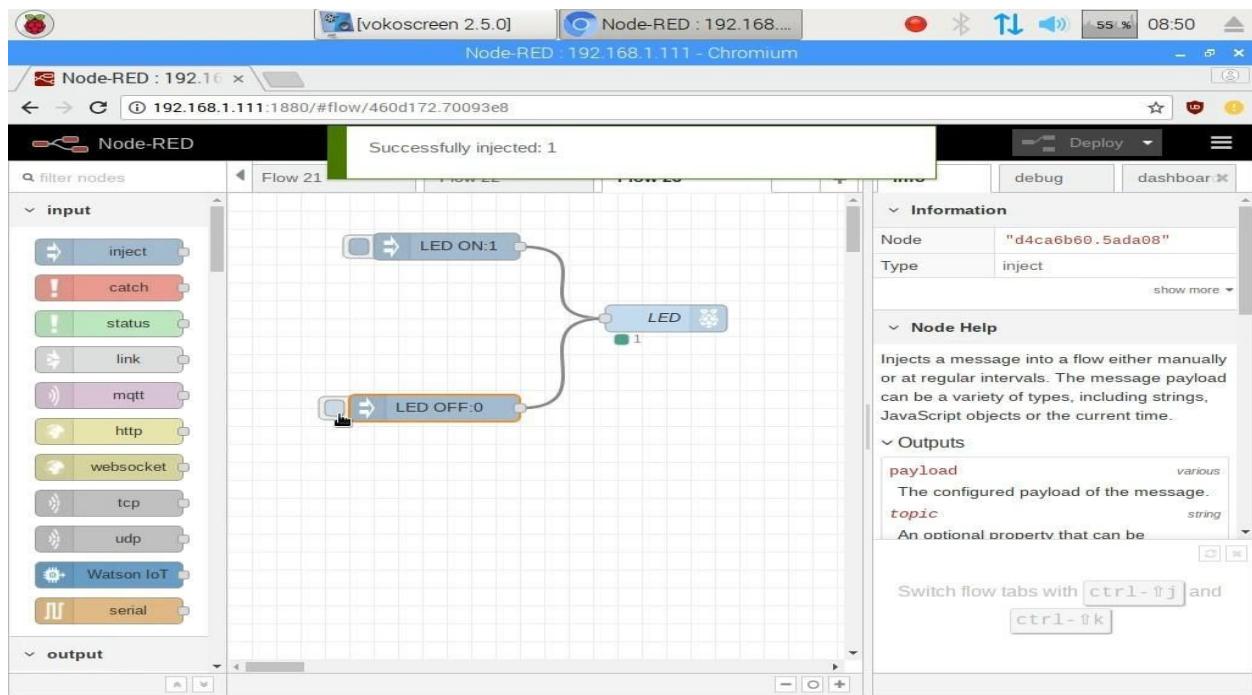
**Step 10 →**

**Now Click on the Deploy at the right top.**

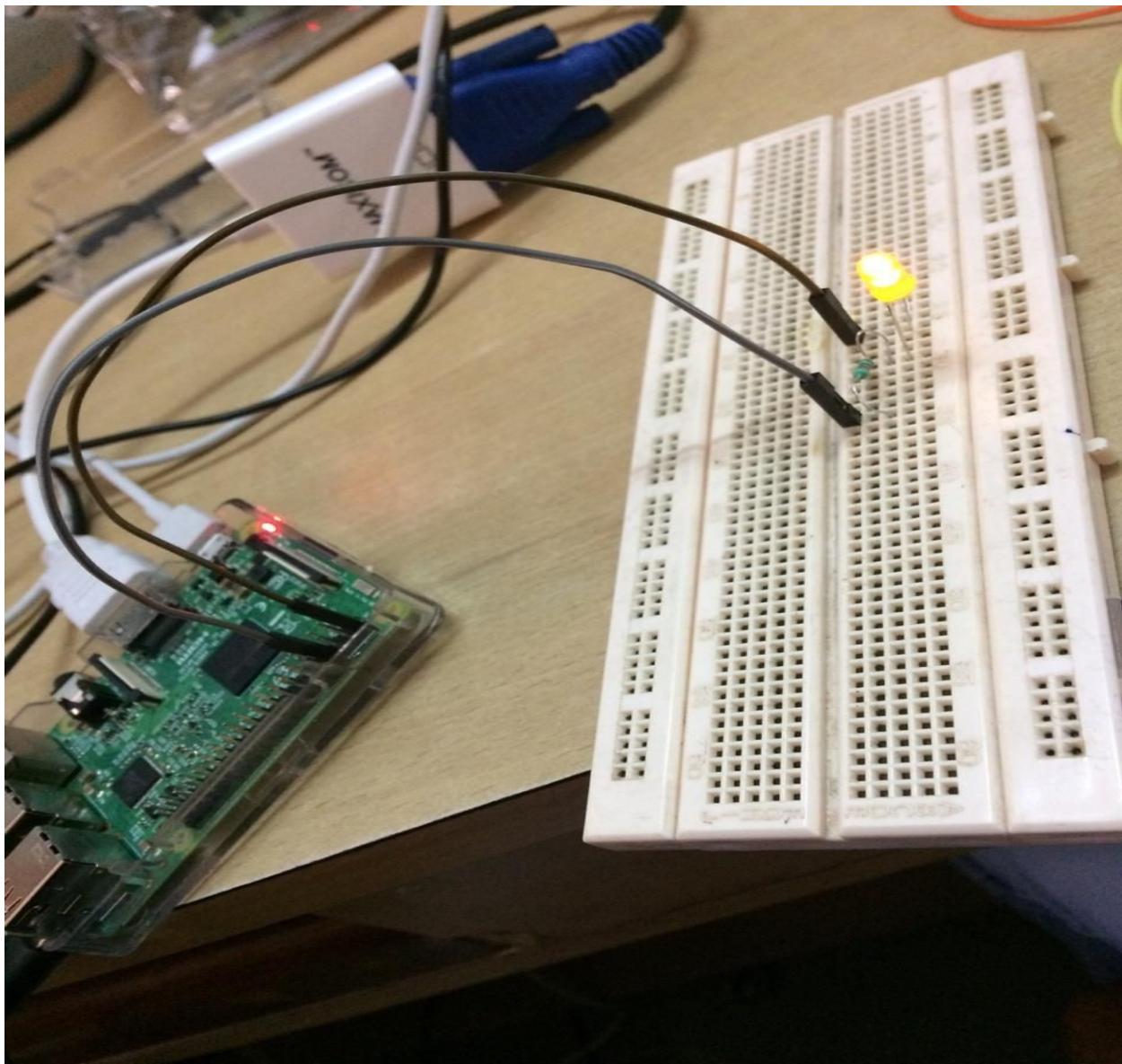


**Step 11 →**

**Click on LED ON:1 for  
blow the LED .  
Click on LED OFF:0  
for Shut the LED .**

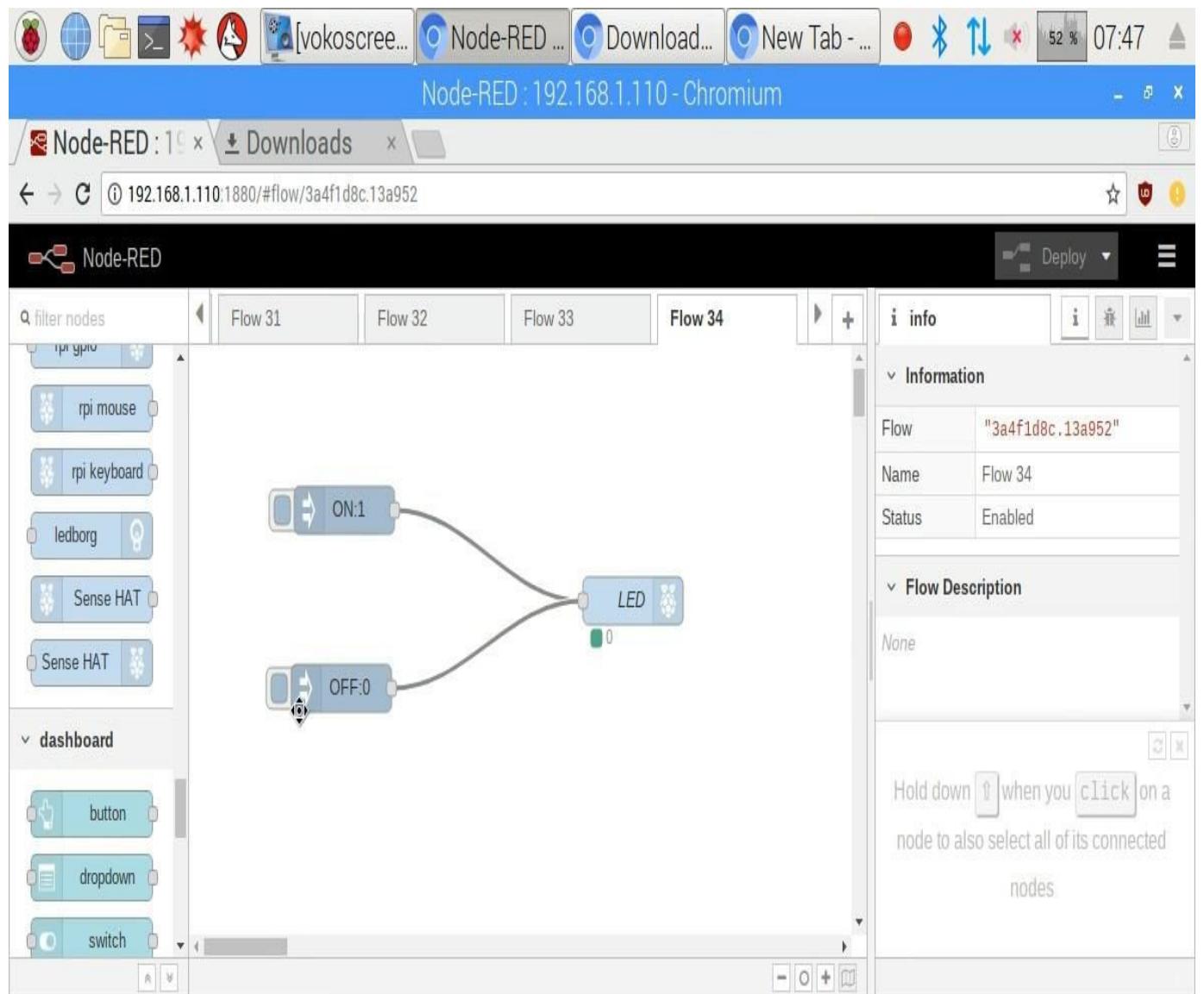


## HARDWARE COMPONENT

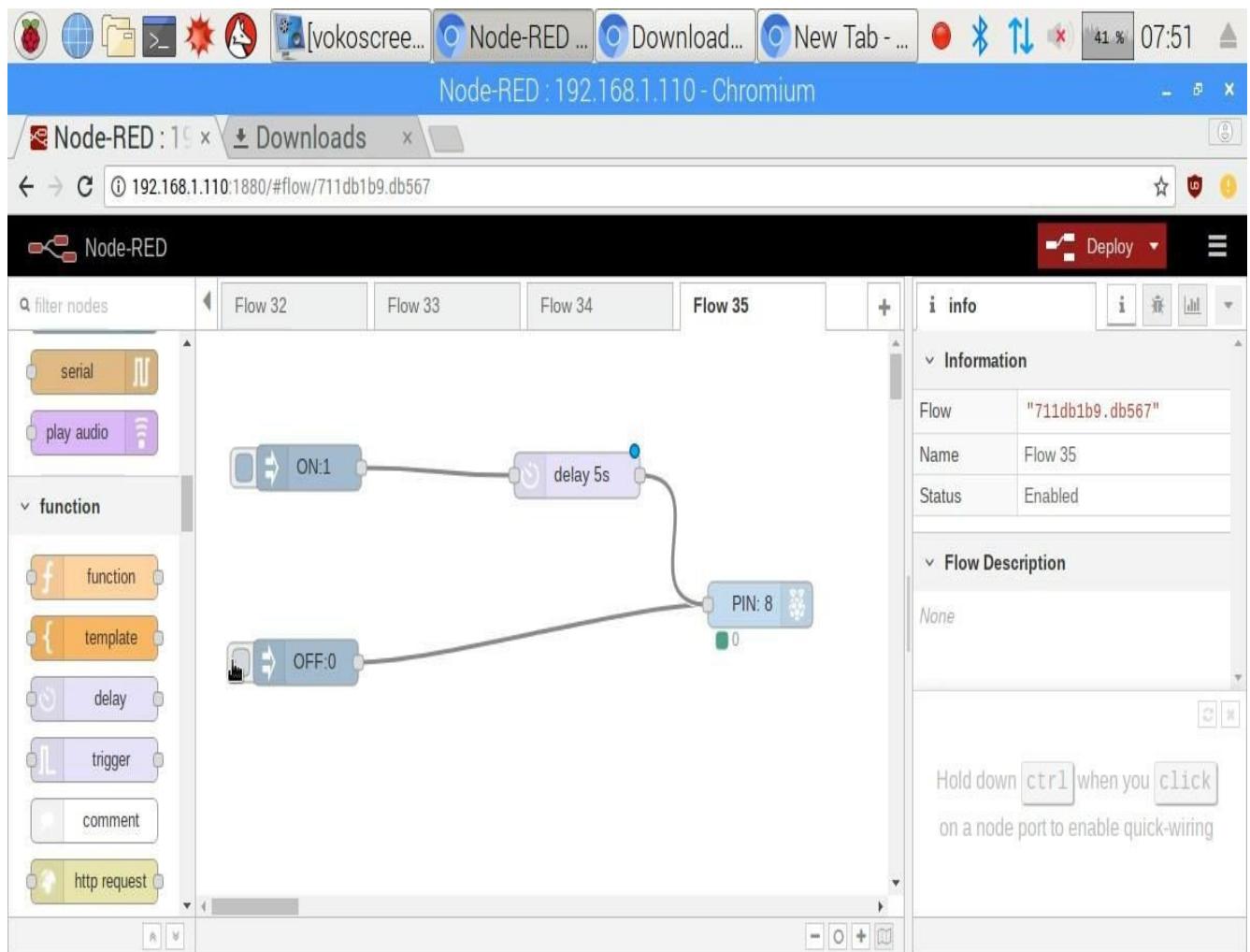


**Following are the examples of the  
programs performed using Node-RED**

## 1. Glowing up a LED using Node-RED.



## 2. Glowing up a LED with delay using Node-RED.



## PRACTICAL NO . 8

### Raspberry Pi Programming Using Online Simulator .

#### GPIO LED WITH PYTHON PROGRAMS :

1. Write a program to glow an LED in board mode by using pin number 8.

The screenshot shows the R-Pi Simulator interface. At the top left is the UNED logo and the R-Pi Simulator icon. The code editor window contains a file named 'mycode.py' with the following content:

```
1 import RPi.GPIO as GPIO
2 GPIO.setwarnings(False)
3 GPIO.setmode(GPIO.BOARD)
4 GPIO.setup(8,GPIO.OUT)
5 GPIO.output(8,False)
6 while True:
7     GPIO.output(8,True)
```

Below the code editor is a window titled 'mycode.py' showing the 'RPi GPIO connectors'. The connectors are labeled with pin numbers and their functions. Pin 8 is highlighted with a red dot. The window also includes a 'Stop' button and a 'Hide' button. At the bottom right, there is a message: 'Limiting speed to avoid crashing the browser: 1000 times per second' with icons for play, stop, and settings.

2. Write a program to glow an LED in board mode by using any pin number.

The screenshot shows the R-Pi Simulator interface. At the top left is the UNED logo and the R-Pi Simulator icon. Below is a code editor window titled "mycode.py" containing the following Python code:

```
1 import RPi.GPIO as GPIO
2 GPIO.setwarnings(False)
3 GPIO.setmode(GPIO.BCM)
4 GPIO.setup(22,GPIO.OUT)
5 GPIO.output(22,False)
6 while True:
7     GPIO.output(22,True)
```

Below the code editor is a window titled "mycode.py" displaying the text "RPi GPIO connectors:" above a diagram of the Raspberry Pi's GPIO pins. The diagram shows two rows of pins, labeled 1 through 40. Pin 22 is highlighted with a red dot, indicating it is the active output pin. The pins are color-coded: red for 5v Power, green for Ground, and various colors for the GPIO pins. Below the diagram are two buttons: "Stop" and "Hide". At the bottom of the window is a message: "Limiting speed to avoid crashing the browser: 1000 lines per second".

3. Write a program to glow an LED at pin number of 36 Board mode.

UNED R-Pi Simulator

mycode.py

```
1 import RPi.GPIO as GPIO
2 GPIO.setwarnings(False)
3 GPIO.setmode(GPIO.BOARD)
4 GPIO.setup(36,GPIO.OUT)
5 GPIO.output(36,False)
6 while True:
7     GPIO.output(36,True)
```

mycode.py

RPi GPIO connectors:

Exception: Cannot set output for input pin 36 (BCM 16) on line 7

Stop Hide

Limiting speed to avoid crashing the browser: 1000 times per second

4. Write a program to glow the LED by accepting pin number from the user in BOARD mode.

```
mycode.py
1
2 import RPi.GPIO as GPIO
3 GPIO.setwarnings(False)
4 GPIO.setmode(GPIO.BCM)
5 x=int(input('Enter a pin number:'))
6 GPIO.setup(x,GPIO.OUT)
7 GPIO.output(x,False)
8 while True:
9     GPIO.output(x,True)
10
11
```

mycode.py

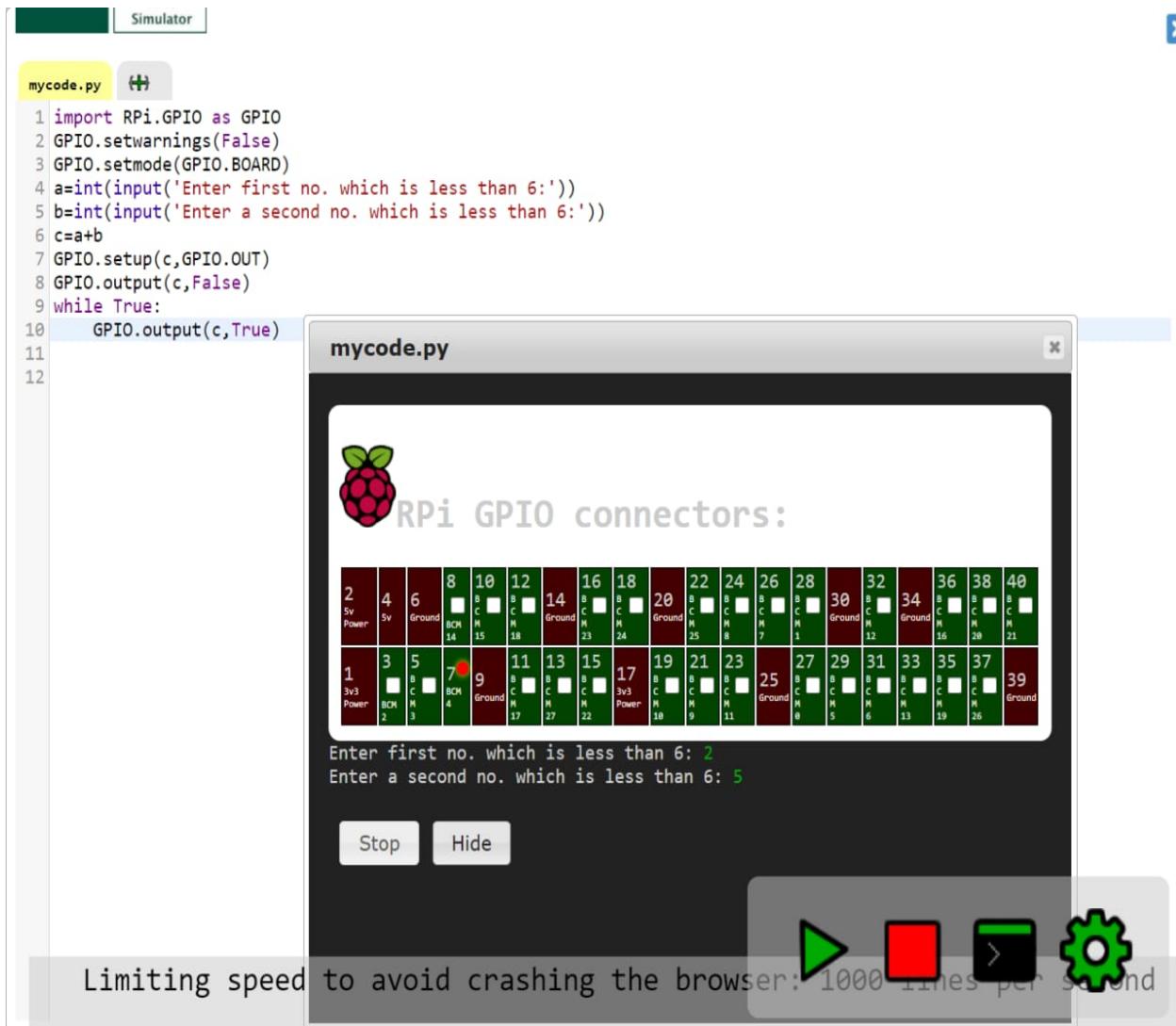
RPi GPIO connectors:

Enter a pin number: 16

Stop Hide

Limiting speed to avoid crashing the browser: 1000 times per second

5. Write a program to glow the LED by accepting any 2 number from the user (number should be less than 6). Add 2 nos and result of addition should be your pin no. to glow an LED.



```

mycode.py + Simulator
mycode.py

1 import RPi.GPIO as GPIO
2 GPIO.setwarnings(False)
3 GPIO.setmode(GPIO.BCM)
4 a=int(input('Enter first no. which is less than 6:'))
5 b=int(input('Enter a second no. which is less than 6:'))
6 c=a+b
7 GPIO.setup(c,GPIO.OUT)
8 GPIO.output(c,False)
9 while True:
10    GPIO.output(c,True)
11
12

```

**RPi GPIO connectors:**

2 5v Power	4	6 Ground	8 B C M 14	10 B C M 15	12 B C M 18	14 Ground	16 B C M 23	18 B C M 24	20 Ground	22 B C M 25	24 B C M 7	26 B C M 8	28 B C M 1	30 Ground	32 B C M 12	34 Ground	36 B C M 16	38 B C M 20	40 B C M 21
1 3v3 Power	3	5	7 Ground	9 B C M 4	11 B C M 17	13 B C M 27	15 B C M 22	17 3v3 Power	19 B C M 18	21 B C M 9	23 B C M 11	25 Ground	27 B C M 8	29 B C M 5	31 B C M 6	33 B C M 13	35 B C M 19	37 B C M 26	39 Ground

Enter first no. which is less than 6: 2  
Enter a second no. which is less than 6: 5

Stop Hide

Limiting speed to avoid crashing the browser: 1000 lines per second

6. Write a program to blink an Led by accepting the pin no and sleep time from user.

```
mycode.py
1 import RPi.GPIO as GPIO
2 import time
3 GPIO.setwarnings(False)
4 GPIO.setmode(GPIO.BCM)
5 x=int(input('Enter a board number='))
6 GPIO.setup(x,GPIO.OUT)
7 GPIO.output(x,False)
8 while True:
9     GPIO.output(x,True)
10    time.sleep(1)
11    GPIO.output(x,False)
12    time.sleep(1)
```

RPi GPIO connectors:

2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38	40
5v Power	5v	Ground	BCM 14	BCM 15	BCM 18	Ground	BCM 23	BCM 24	Ground	BCM 25	BCM 26	BCM 27	BCM 28	Ground	BCM 31	Ground	BCM 36	BCM 38	BCM 40
1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31	33	35	37	39
3v3 Power	BCM 2	BCM 3	BCM 4	Ground	BCM 17	BCM 18	BCM 22	3v3 Power	BCM 10	BCM 9	BCM 11	BCM 8	BCM 5	BCM 6	BCM 13	BCM 19	BCM 26	BCM 28	Ground

Enter a board number= 31

Stop Hide

▶ ⚡ > ⚙

7. Write a program to glow any two LED's in BOARD mode.

Simulator

mycode.py

```
1 import RPi.GPIO as GPIO
2 GPIO.setwarnings(False)
3 GPIO.setmode(GPIO.BCM)
4 GPIO.setup(15,GPIO.OUT)
5 GPIO.setup(18,GPIO.OUT)
6 GPIO.output(15,False)
7 GPIO.output(18,False)
8 while True:
9     GPIO.output(15,True)
10    GPIO.output(18,True)
```

mycode.py

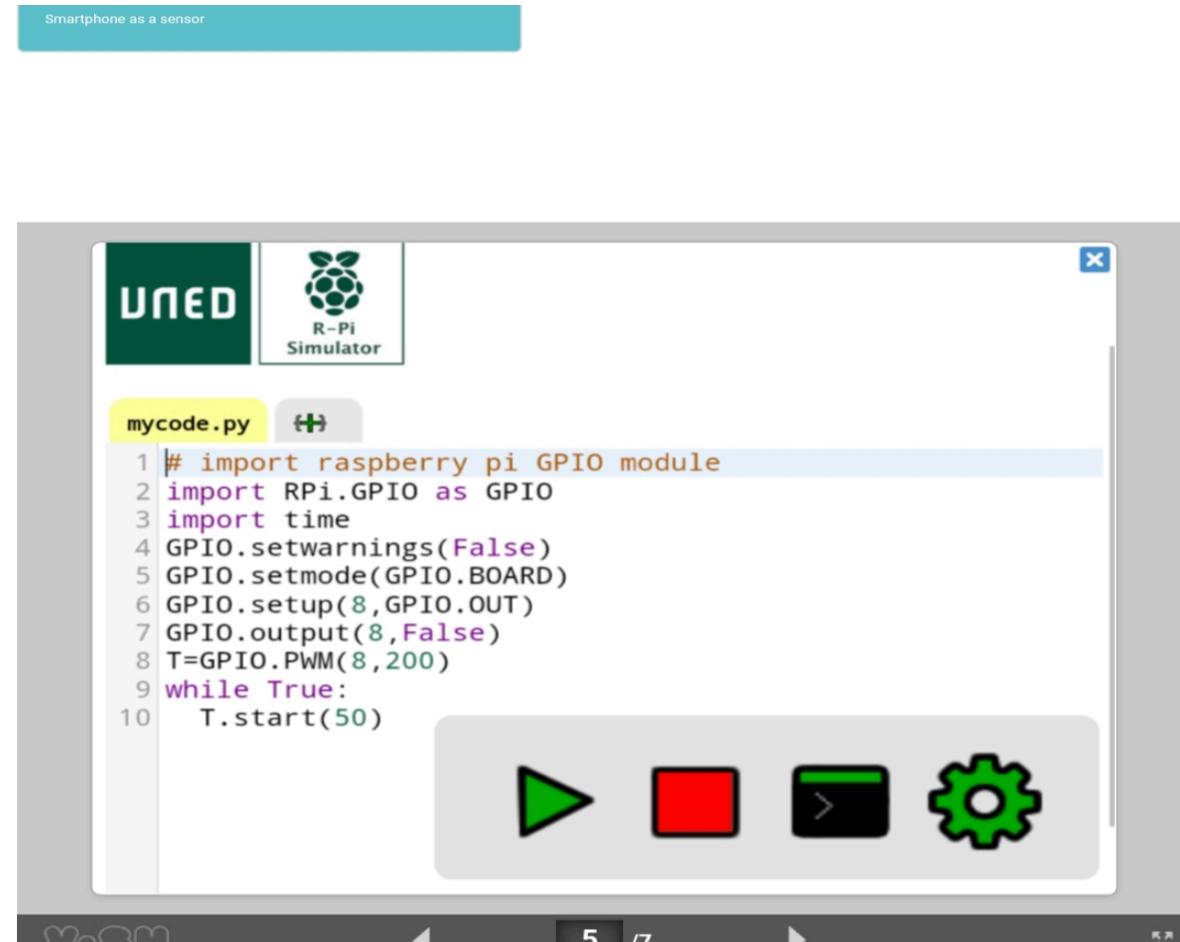
RPi GPIO connectors:

Stop Hide

Limiting speed to avoid crashing the browser: 1000 times per second

## PWM PROGRAMS :

1. WRITE A PROGRAM TO INCREASE THE INTENSITY OF AN LED BY USING PWM.



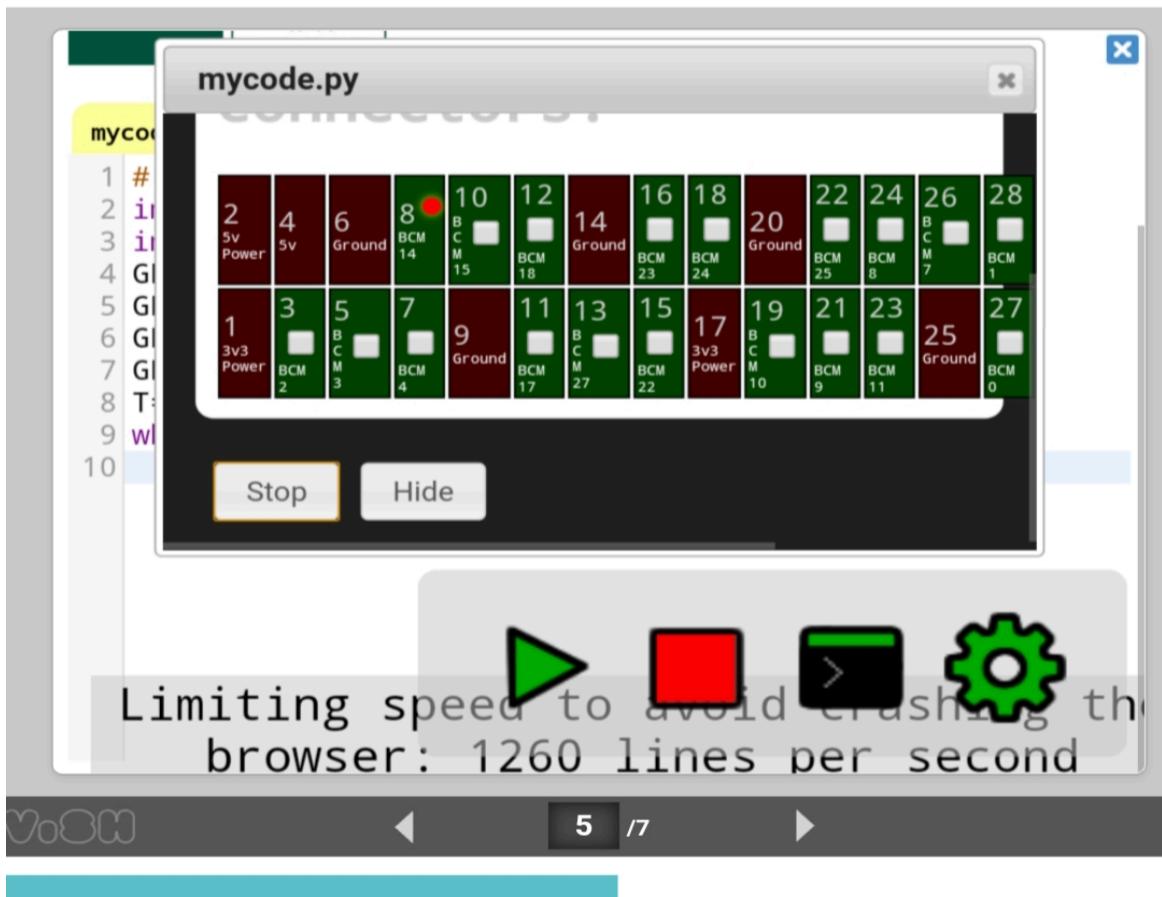
The screenshot shows the R-Pi Simulator software interface. At the top, there's a teal bar with the text "Smartphone as a sensor". Below it is a window titled "mycode.py" which contains the following Python code:

```
1 # import raspberry pi GPIO module
2 import RPi.GPIO as GPIO
3 import time
4 GPIO.setwarnings(False)
5 GPIO.setmode(GPIO.BCM)
6 GPIO.setup(8,GPIO.OUT)
7 GPIO.output(8,False)
8 T=GPIO.PWM(8,200)
9 while True:
10     T.start(50)
```

Below the code editor are four large buttons: a green play button, a red square, a black square with a white right-pointing arrow, and a green gear icon. At the bottom of the window, there's a toolbar with icons for file operations and a status bar showing "5 /7".

**Internet of Things for European Small and Medium Enterprises ©**  
*Erasmus+ Project N. 2016-1-IT01-KA202-005561*  
[Privacy statement and cookie policy](#)

## OUTPUT :



2. WRITE A PROGRAM TO INCREASE THE INTENSITY OF AN LED BY USING FOR LOOP IN PWM .



mycode.py

```
1 # import raspberry pi GPIO module
2 import RPi.GPIO as GPIO
3 import time
4 GPIO.setwarnings(False)
5 GPIO.setmode(GPIO.BRD)
6 GPIO.setup(8,GPIO.OUT)
7 GPIO.output(8,False)
8 T=GPIO.PWM(8,50)
9 while True:
10     for i in range(0,100):
11         T.start(i)
12         time.sleep(0.05)
```

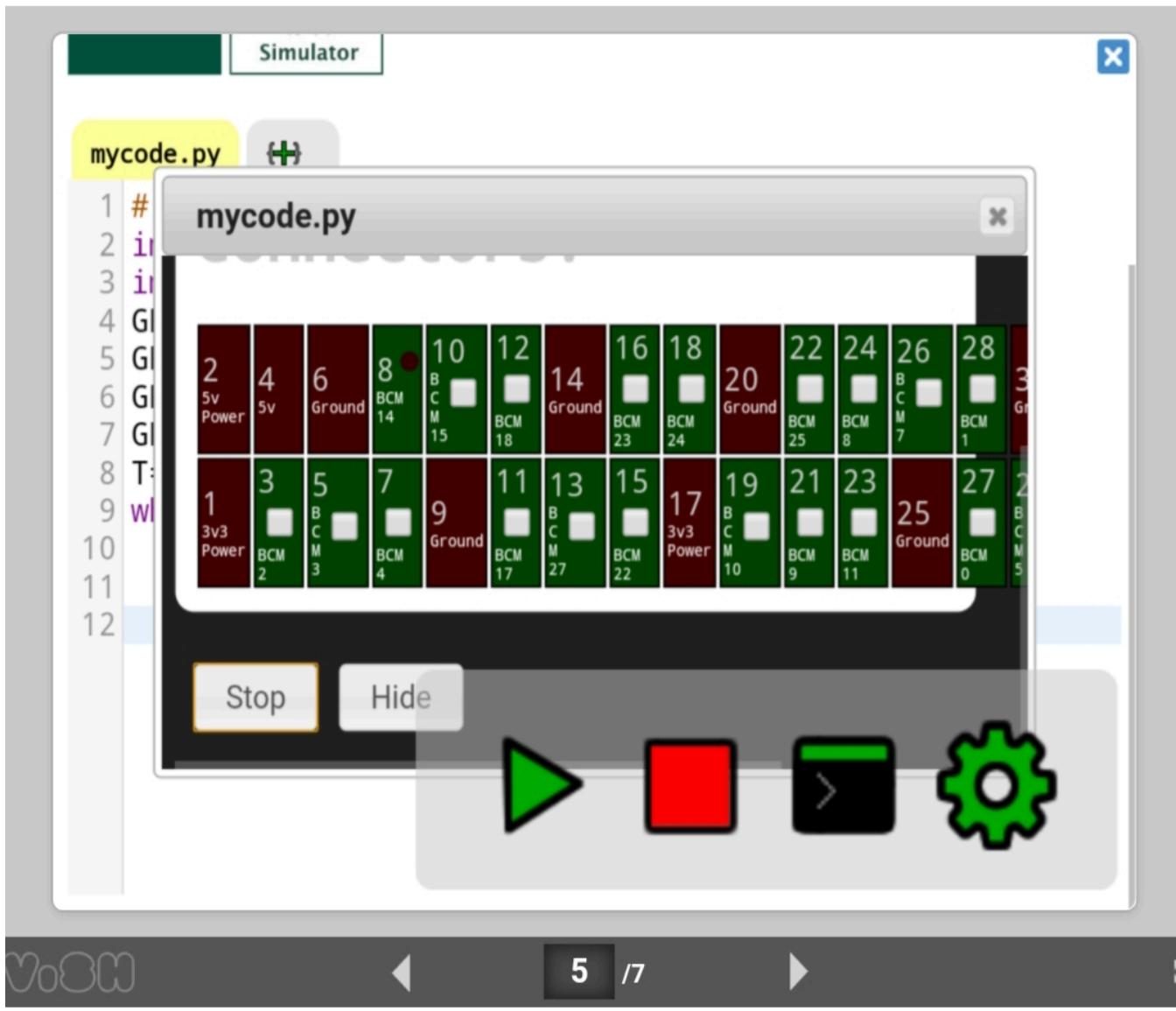
Simulator

VoOB

5 /7

Internet of Things for European Small and

## OUTPUT :



## PRACTICAL NO.9

### STUDY OF XMPP SERVER PROTOCOL

STEP 1 :

Download the XMPP app on your device .

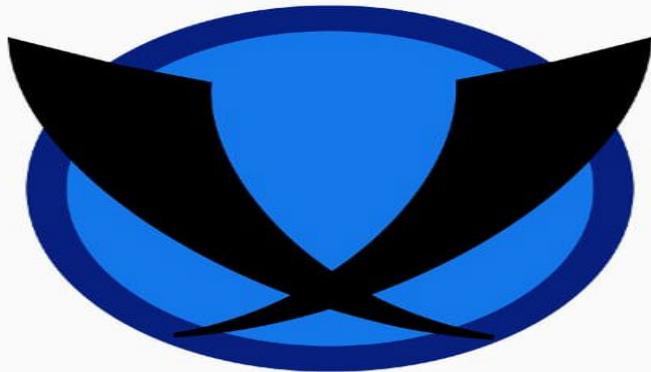


Step 2 :

Create an account on the app which is also called as JABBER ID .



## Create Account



### Pick your username

We will guide you through the process of creating an account on conversations.im.<sup>1</sup> When picking conversations.im as a provider you will be able to communicate with users of other providers by giving them your full Jabber ID.

Username

NEXT

Step 3 :

Set an password for your JABBER ID .

**Create Account**

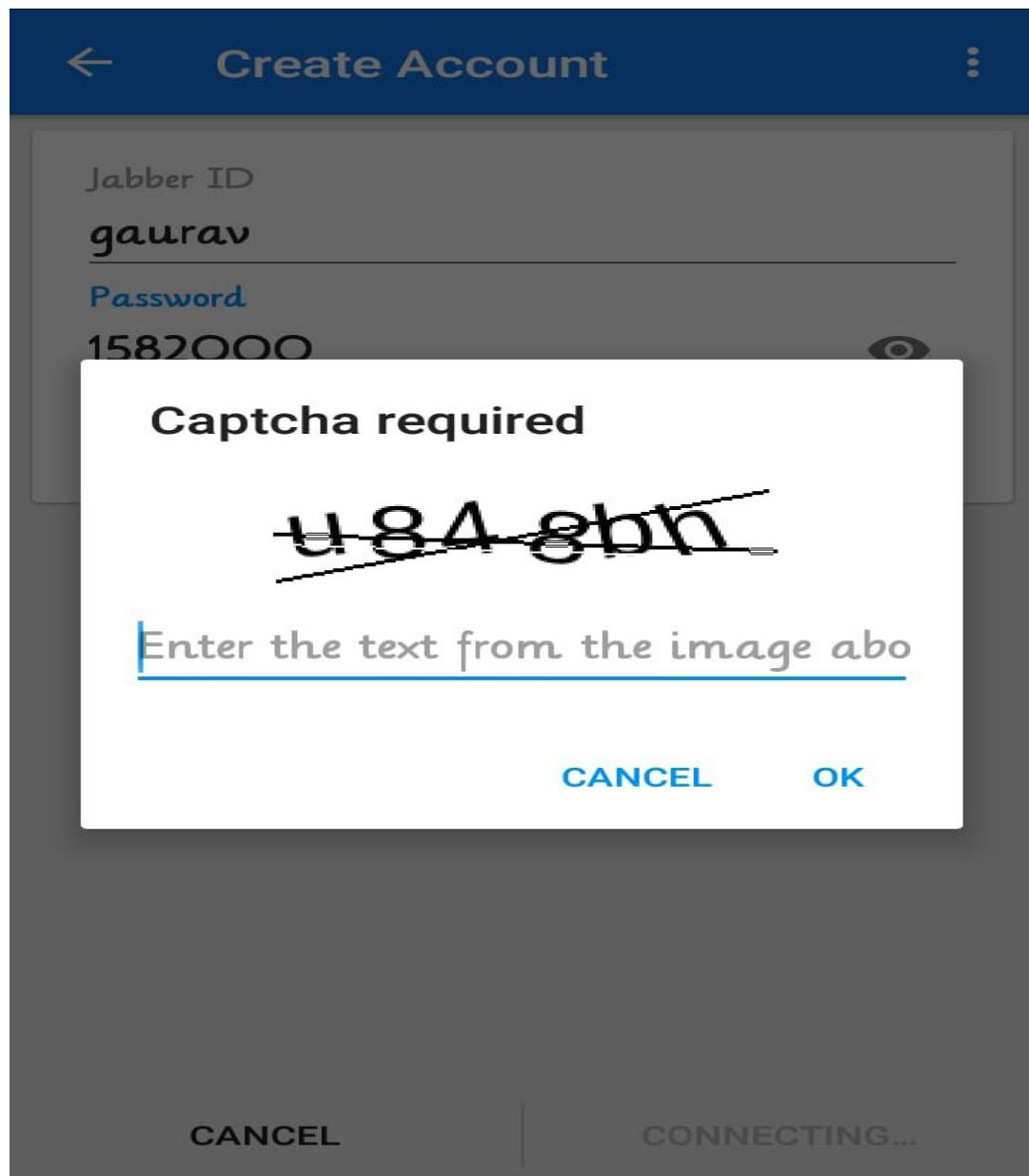
**Jabber ID**  
gaurav

**Password**

**CANCEL** | **NEXT**

Step 4 :

Enter the Captcha Code mentioned in the page.



Step 5 :

Set Avatar to publish of your choice .

Or long press for default avatar .



## Publish avatar



Touch avatar to select picture from gallery  
(Or long press to bring back default)

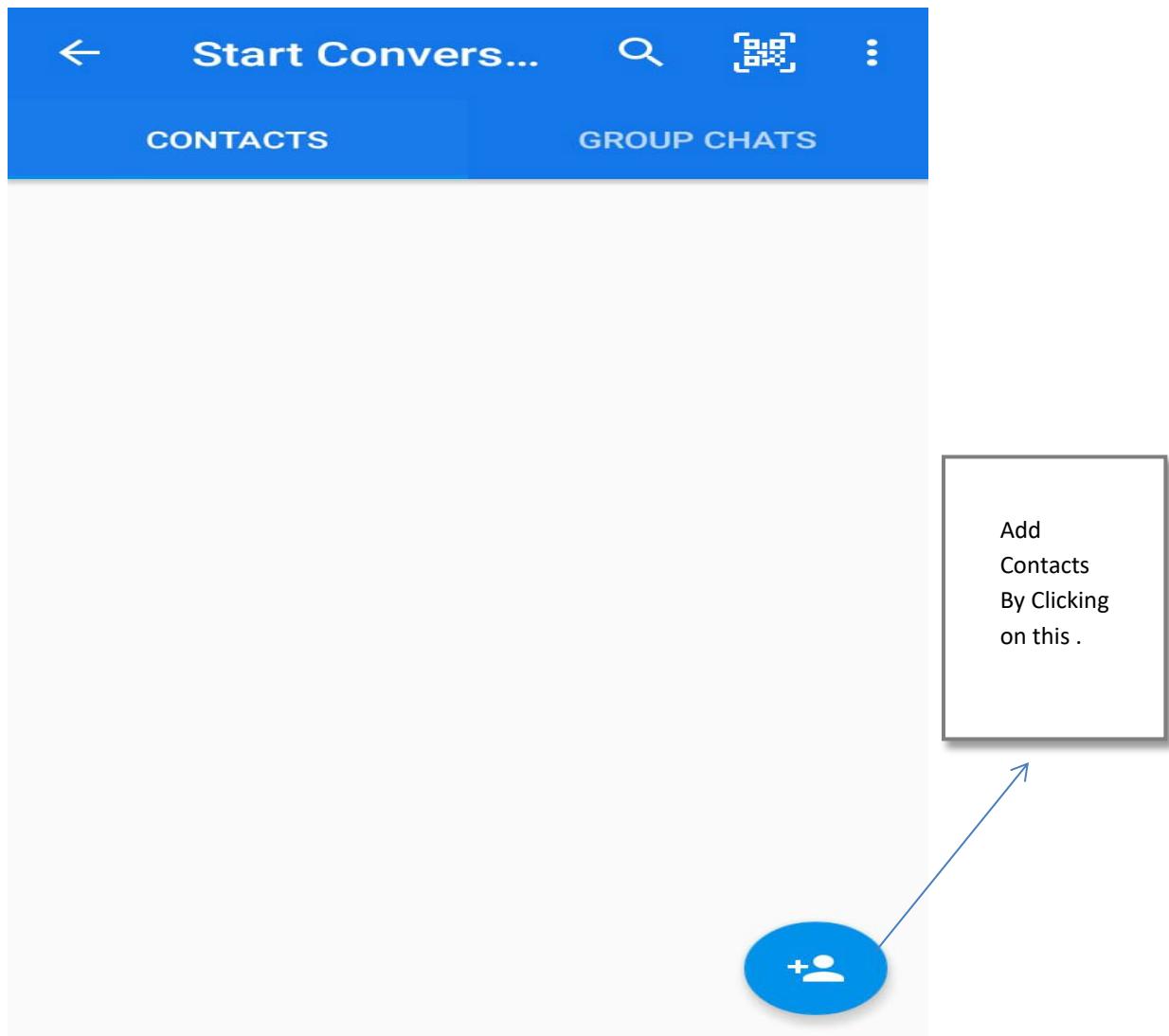
SKIP

PUBLISH

Step 6 :

Your account is created and ready to use .

And now can add others to chat in app .



### Step 7 :

You can do various settings like encryption , notifications .

By clicking on settings in the app .



## Settings

### General

#### Share XMPPJabberClient

Share the App with your Friends.

#### Rate XMPPJabberClient

Please take a moment to rate our application and service on play store.

### Privacy

#### OMEMO Encryption

OMEMO will have to be turned on explicitly for new conversations.

#### Confirm Messages

Let your contacts know when you have received and read their messages



#### Typing notifications

Let your contacts know when you



Step 8 :

Now you can message on the app by adding anyone you want to talk .

## XMPPJabberClient



S

shylashreedev

just now

Me: Hello

C

conversations.im

4 mins ago

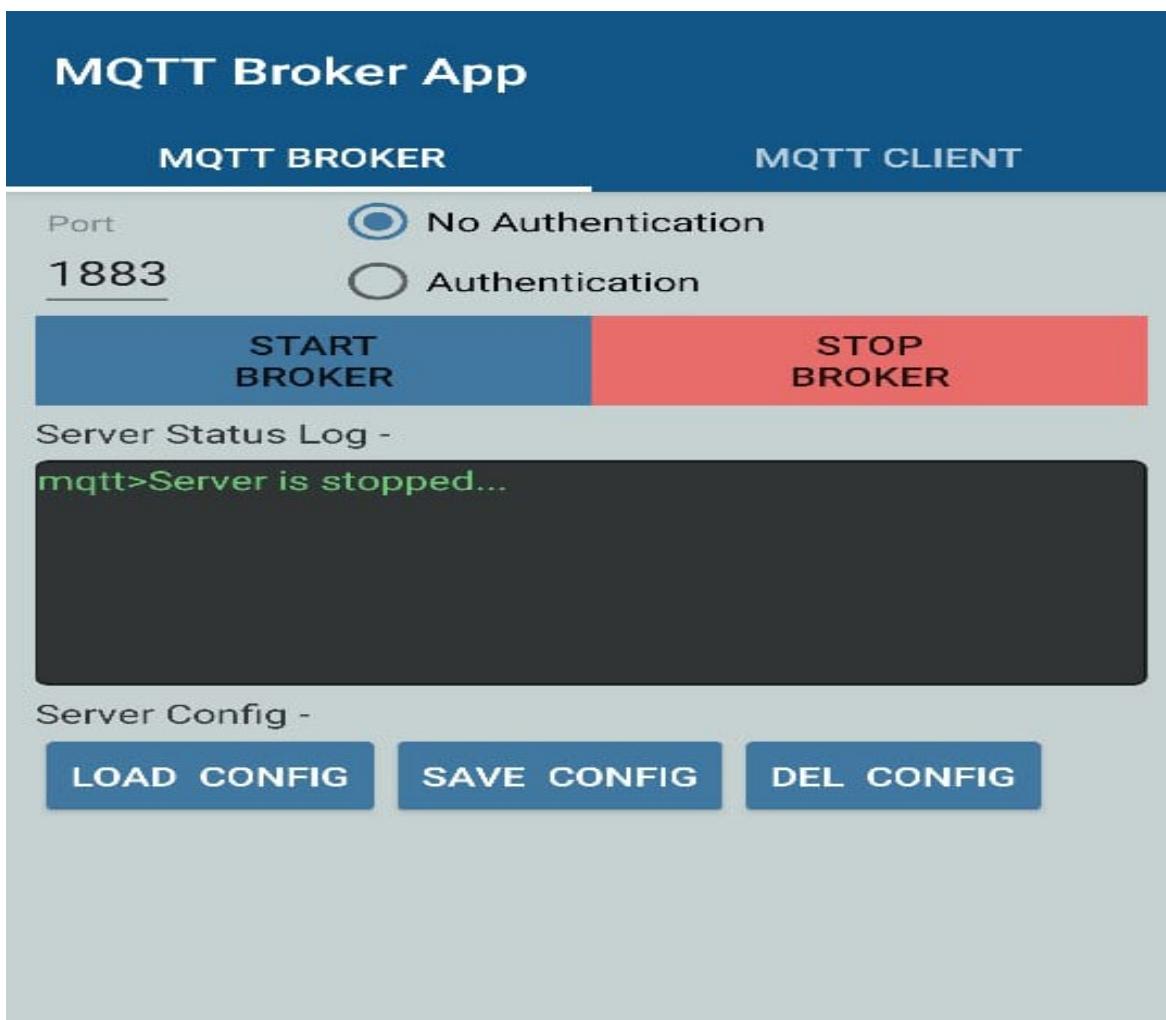
Thank you for choosing conversation...



**PRACTICAL NO. 10**  
**STUDY OF MQTT PROTOCOL**

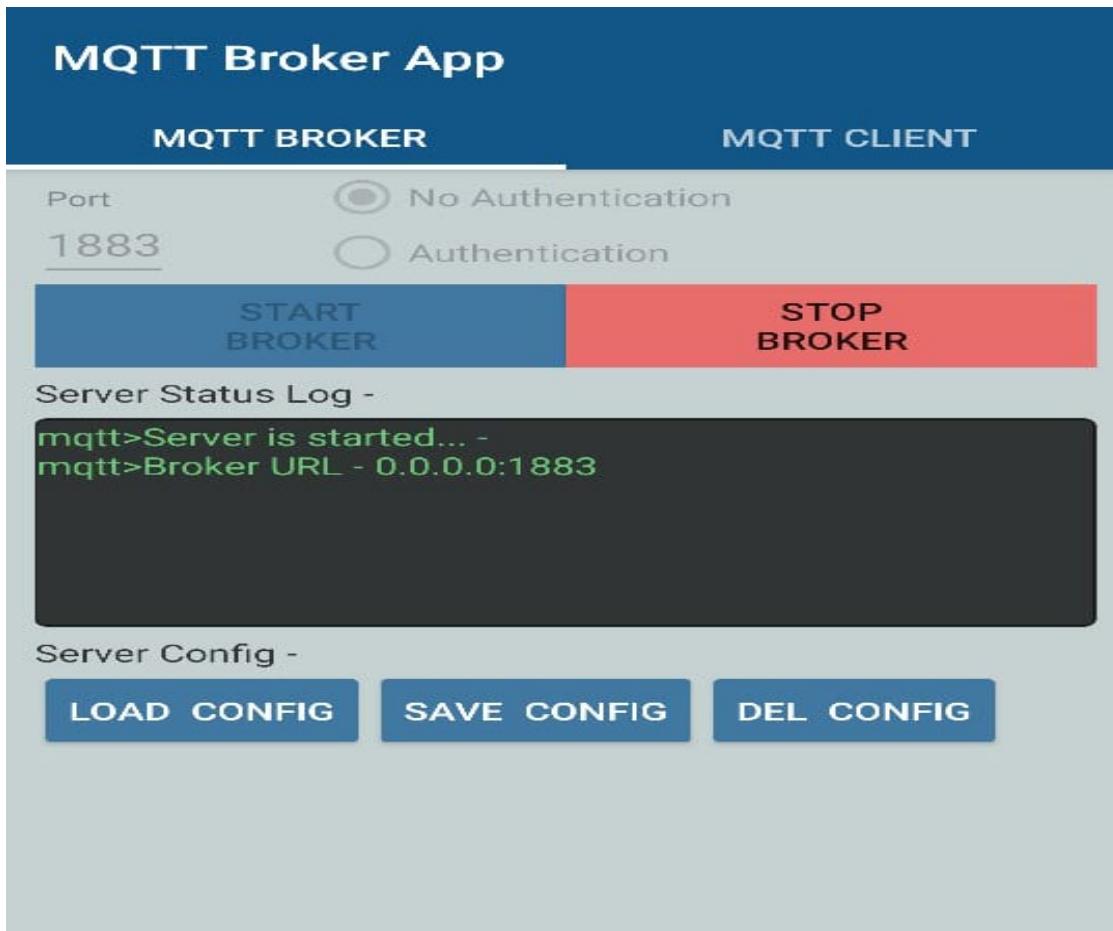
Step 1 :

Download the MQTT app on your device .



Step 2 :

Click on **start broker** in the page .



Step 3 :

Goto **MQTT Client** and click on top corner symbol and create an **client id** for protocol

**Create Mqtt Client**

Enter Values Below -

**Client ID**

No Authentication  
 Authentication

**Choose Broker**

**CREATE**

Choose Broker as **Local Host** Broker .

**Create Mqtt Client**

Enter Values Below -

**Client ID**

Hello

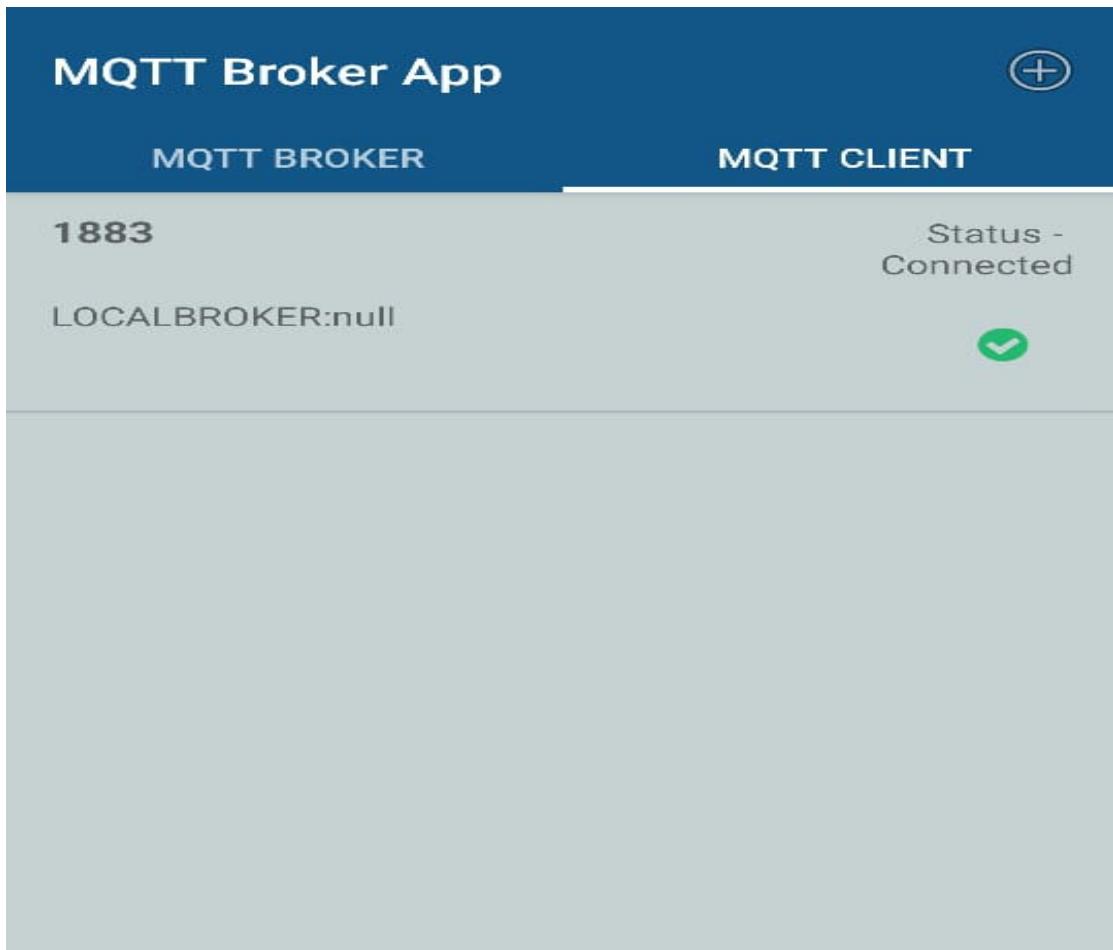
No Authentication  
 Authentication

**LocalHost Broker**

**CREATE**

Step 4 :

Client Id gets created and shows status as connected .



Step 5 :

Now click on it and Goto **Publish** option page .

Here publish **Name , Value , Topic** .

Enter Values Below -

Enter Publish Name

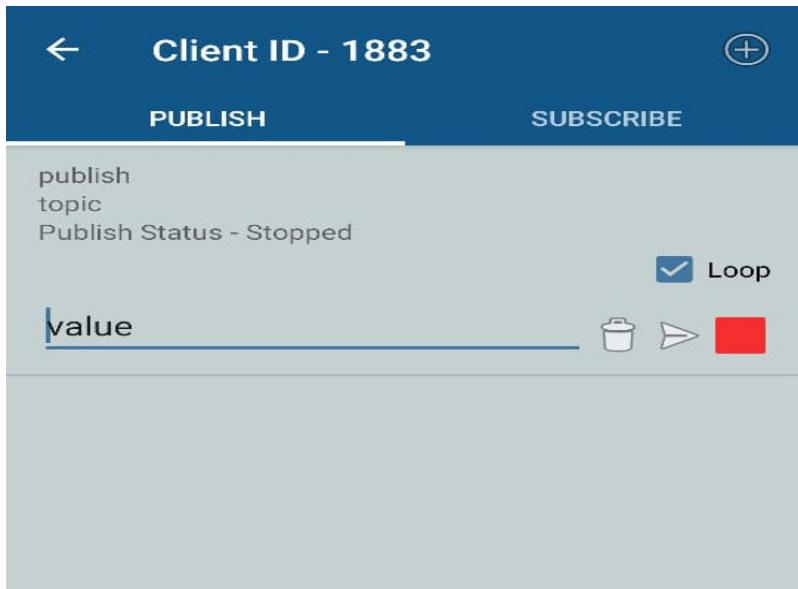
Enter Publish Value

Enter Topic Name

Loop

**ADD**

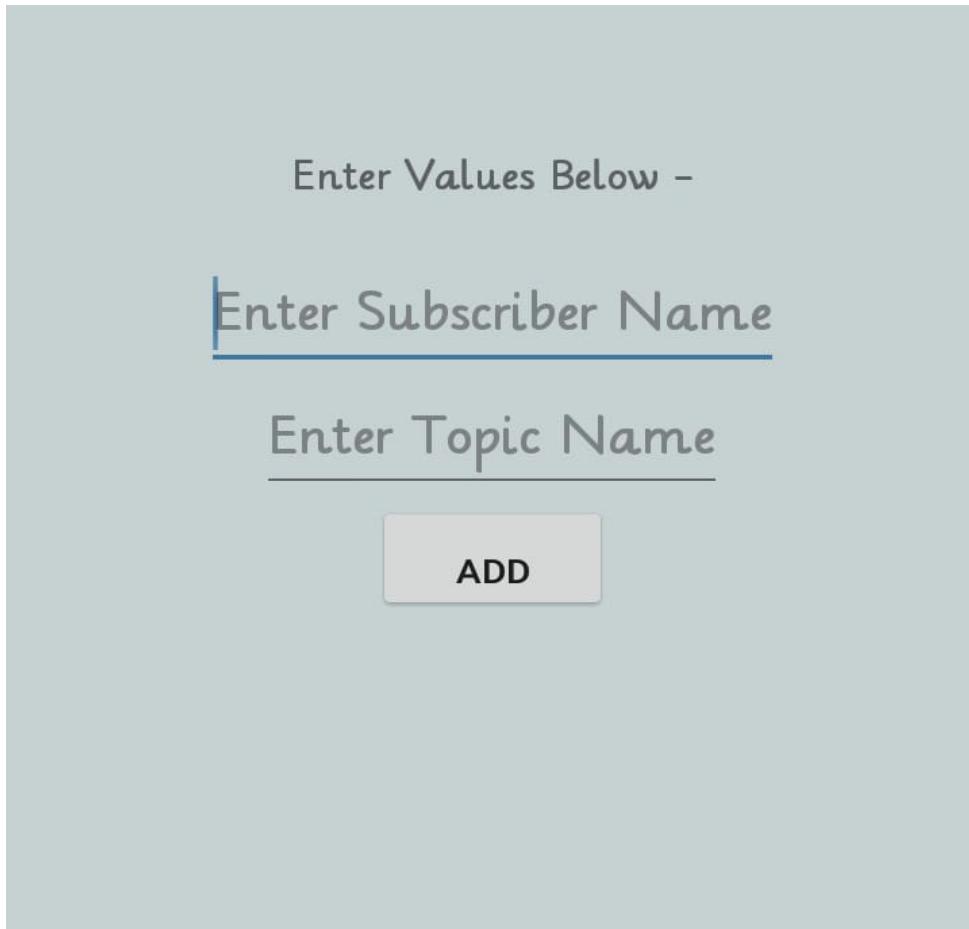
Click on add and then topic gets **published** .



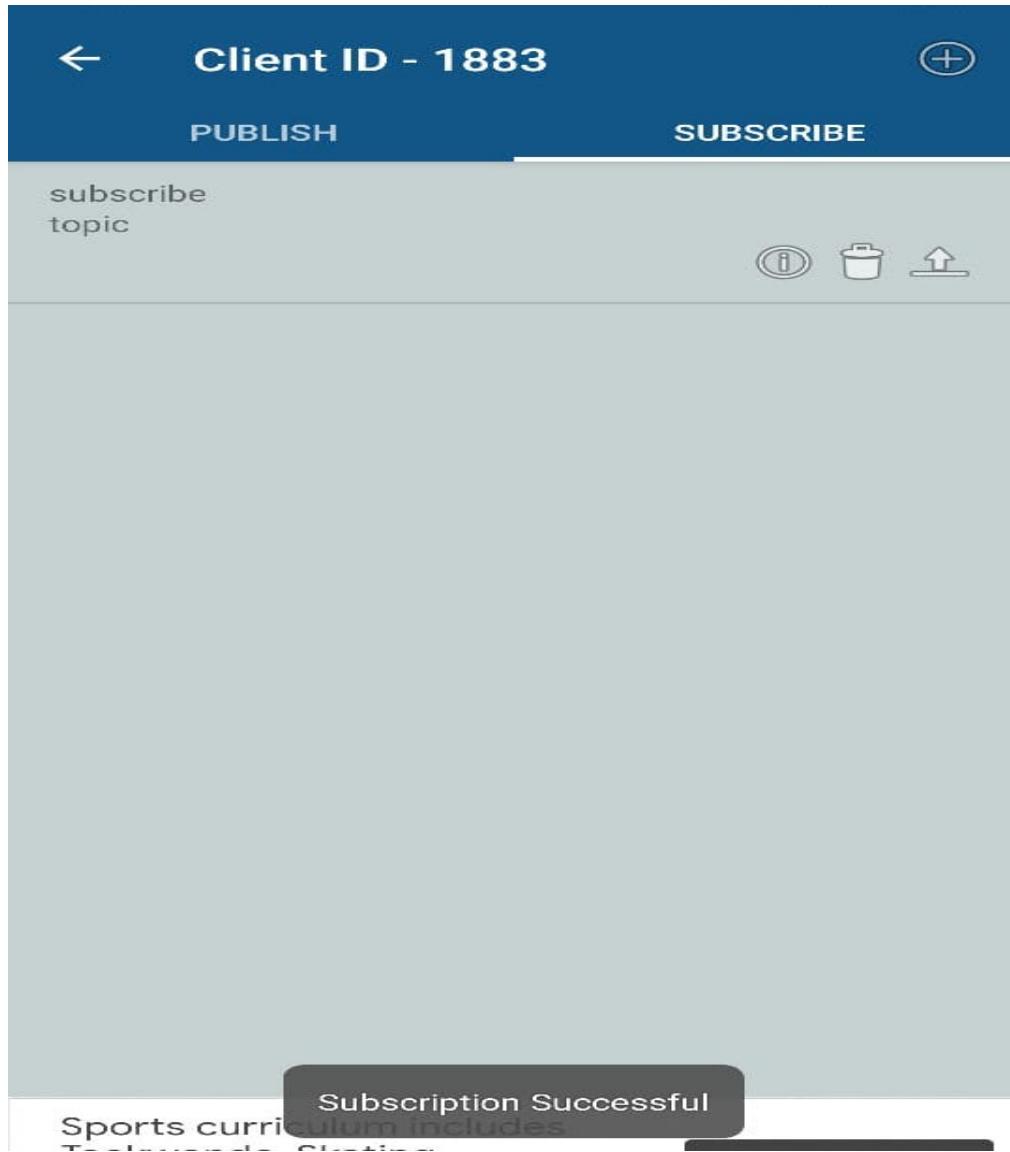
Step 6 :

Now Goto to subscribe option page .

Here enter subscriber name and topics name as given in publish .



Click on add and then the topic gets **subscribed** .



Step 7 :

After subscription is successful we publish the message and it is subscribed via broker and we get an output as :



## Subscribed Topic Message

```
mqtt_topic_topic>
value
value
value
value
```

Subscription Successful