EXPERIMENT NUMBER : 3
EXPERIMENT NAME: WEB SERVER IMPLEMENTATION USING RASPBERRY PI
DATE: 03/11/2022, THURSDAY

* <u>AIM</u>:
Creation of Web server and its interactions using Flask Framework on Raspberry Pi with SenseHAT.

* <u>INTEGRATED DEVELOPMENT ENVIRONMENT</u>:
Name - Thonny 4.0.1
Publisher - Aivar Annamaa
Support link - https://thonny.org

(a) Web server Implementation using Raspberry Pi
→ <u>Algorithm</u>-
① From sense_hat import SenseHAT python module to control the Raspberry Pi Sense HAT.
② No arguments defaults to OFF and toggle the LED matrix in low light mode, useful if the Sense HAT is being used in a dark environment.
③ From flask import Flask, lightweight Web Server Gateway Interface (WSGI) web application framework.
④ Flask constructor takes the name of current module (__name__) as argument.
⑤ Initialize the colours and define the modules for 'Keep Out Set', 'Ok To Come In Set' and 'Off' routes.
⑥ Map the URLs to a specific function that will handle the logic for that URL.
⑦ Run the app in the main function.

$\overrightarrow{PTO}$

→ Python Code -

```
from sense_hat import SenseHat
sense = SenseHat()
sense. clear()
sense. low_light = True

from flask import Flask
app = Flask (__name__)

green = (0, 255, 0)
red = (255, 0, 0)
nothing = (0, 0, 0)

def scene_keep_out():
    R = red
    scene = [
        R, R, R, R, R, R, R, R,
        R, R, R, R, R, R, R, R,
        R, R, R, R, R, R, R, R,
        R, R, R, R, R, R, R, R,
        R, R, R, R, R, R, R, R,
        R, R, R, R, R, R, R, R,
        R, R, R, R, R, R, R, R,
        R, R, R, R, R, R, R, R]
    return scene

def scene_come_in():
    G = green
    scene = [
        G, G, G, G, G, G, G, G,
        G, G, G, G, G, G, G, G,
        G, G, G, G, G, G, G, G,
        G, G, G, G, G, G, G, G,
        G, G, G, G, G, G, G, G,
        G, G, G, G, G, G, G, G,
        G, G, G, G, G, G, G, G,
        G, G, G, G, G, G, G, G]
    return scene


def scene_off():
    N = nothing
    scene = [
        N, N, N, N, N, N, N, N,
        N, N, N, N, N, N, N, N,
        N, N, N, N, N, N, N, N,
        N, N, N, N, N, N, N, N,
        N, N, N, N, N, N, N, N,
        N, N, N, N, N, N, N, N,
        N, N, N, N, N, N, N, N,
        N, N, N, N, N, N, N, N]
    return scene

@app.route('/')
def hello_world():
    return 'Lighting Scene Controller!'

@app.route('/keep-out')
def keep_out():
    sense.set_pixels(scene_keep_out())
    return 'Keep Out Set'

@app.route('/come-in')
def come_in():
    sense.set_pixels(scene_come_in())
    return 'Ok To Come In Set'
```

→ OUTPUTS :

CASE I : 172.17.126.47 :5000/
Lighting Scene Controller !

CASE II : 172.17.126.47: 5000/Keep out
Keep out Set
} Red LED matrix

CASE III : 172.17.126.47: 5000/come-in
Ok To Come In Set
} Green LED matrix

CASE IV : 172.17.126.47: 5000/off
off
} Default LED matrix (nothing)

```
@app.route ('/off')
def off ():
    sense.set_pixels (scene_off ())
    return 'off'

# pi@raspberrypi:~ $ ifconfig
# eth0 : flags = 4163 <UP, BROADCAST, RUNNING, MULTICAST> mtu 1500
#          inet 172.17.126.47 netmask 255.255.255.0 broadcast --
#          inet6 fe80::9d5a:ccdd:c209:ebab prefixlen 64 --.
#          ....
#          ....
#          ....
#        TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

# Driver Code - Main Function:-
if __name__ = '__main__':
    app.run (host = '172.17.126.47') # Port Number - 5000
```

(b) **Web server based Weather Station using Raspberry Pi with SenseHat**

→ Algorithm -

① From sense_hat import SenseHat, python module to control the Raspberry Pi Sense HAT.

② No arguments defaults to OFF. Toggle the LED matrix in low light mode, useful if the Sense HAT is being used in a dark environment.

③ From flask import Flask, a lightweight Web Server Gateway Interface (WSGI) web application framework.

④ From flask import render_template, used to generate output from a template file that is found in the application's templates folder.

⑤ The flask constructor takes the name of current module [__name__] as argument.

PTO →

⑥ Import time, the python module providing various time-related functions. Initialize the text and background colours for the LED matrix display.

⑦ Map the URLs to a specific function that will handle the logic for that URL.

⑧ Gets the current pressure in Millibars from the pressure sensor and the temperature in Degrees from the temperature sensor.

⑨ Return the time in seconds since the epoch as a floating point number and convert the time in seconds since the epoch to a string of a form: 'Sun Jun 20 23:21:05 1993' representing local time.

⑩ Print the message in Thonny shell and show the same in the LED matrix with appropriate scroll speed and suitable text and background colour.

⑪ Run the app in the main function.

→ Python Code -

```
from sense_hat import SenseHat
sense = SenseHat()
sense.clear()
sense.low_light = True

from flask import Flask
from flask import render-template
app = Flask(_name_)

import time

green = (0,255,0)
nothing = (0,0,0)

@app.route('/')
def index():
```

PTO →

```
pressure = sense.get_pressure()
temperature = sense.get_temperature()
seconds = time.time()
local_time = time.ctime(seconds)

print_message = str("Weather Monitoring System!" + "\n" +
"Current Pressure: " + str(round(pressure, 2)) + "millibars" + "\n" +
"Temperature: " + str(round(temperature, 2)) + "degree(s)" + "\n" +
"Local Time: " + str(local_time))

sense.show_message("Current Pressure: {} Temperature: {}
Local Time: {}".format(pressure, temperature, local_time),
                    scroll_speed = 0.005,
                    text_colour = green,
                    back_colour = nothing)

weatherData = {'weather': print_message}
return render_template('index.html', **weatherData)

# pi@raspberrypi:~ $ ifconfig
# eth0: flags=4163 <UP, BROADCAST, RUNNING, MULTICAST> mtu 1500
#          inet 172.17.126.47 netmask 255.255.255.0 ....
#          inet6 fe80::9d5a:cedd:c209:ebab prefixlen 64 ...
#          ....
#          ....
#          ....
#          TX errors 0  dropped 0  overruns 0 carrier 0 ....

# Driver Code :- Main Function :-
if __name__ == '__main__'
    app.run(host = '172.17.126.47') # Port Number - 5000
```

→ 'templates' Folder :- index.html -

```
<html>
    <head>
        <link rel = "stylesheet" href = "/static/style.css" />
    </head>
    <body>
        <h1> Weather: </h1>
        <h2 class = "title"> {{weather}} </h2>
    </body>
</html>
```

→ 'static' Folder :- style.css -

```
.title {
    font-family: Roboto;
    color : #FFFFFF;
}

body {
background: navy;
color : yellow;
font-family: Arial, sans-serif;
text-align: center;
}

h1 {
border-bottom: 1px solid gold;
margin-bottom: 40 px;
padding: 10 px;
}
```

PTO

→ <u>OUTPUTS</u>:

<u>index.html</u> -

weather

{{weather}}

c) Web server based Remote Device Control using Raspberry Pi with Sense HAT :

→ Algorithm -

① From sense-hat import SenseHat, python module to control the Raspberry Pi Sense HAT.

② No argument defaults to OFF and toggle the LED matrix in low light mode, useful if the Sense HAT is being used in a dark environment.

③ From flask import Flask, a lightweight Web Server Gateway Interface (WSGI) web application framework.

④ From flask import render-template, used to generate output from a template file that is found in the application's templates folder.

⑤ Flask constructor takes the name of current module (_name_) as argument.

⑥ Map the URLs to a specific function that will handle the logic for that URL.

⑦ Index () definition -

(i) Set the device status to 'OFF' and show the letter 'I' in the LED matrix.

(ii) Store the device name and status in a dictionary and return the same along with the index.html as parameter in render-template().

⑧ action () definition -

(i) For the required device name and action (ON/OFF), set the corresponding device status and show the message on the LED matrix.

(ii) Store the device name and status in a dictionary and return the same along with the index.html as parameter in render_template ().

⑨ clear() definition -

(i) Set the device status as 'OFF'.

(ii) Clear the Sense HAT display matrix.

(iii) Store the device name and status in a dictionary and return the same along with index.html as parameter in render_template ().

⑩ Run the app in the main function.

→ Python code -

```
from sense-hat import SenseHat
sense = SenseHat()
sense.clear()
sense.low_light = True

from flask import Flask
from flask import render_template
app = Flask(__name__)

@app.route('/')
def index():
    devicests = "OFF"
    sense.show_letter("I")
    templateData = {'device1' : devicests}
    return render_template('index.html', **templateData)
#   return render_template('index.html', **weatherData)

@app.route("/<deviceName>/<action>")
def action(deviceName, action):
    if deviceName == 'd1' and action == 'on':
        devicests = "ON"
        sense.show_letter("O")  # Up Arrow
    if deviceName == 'd1' and action == 'off':
        devicests = "OFF"
        sense.show_letter("F")  # Up Arrow
    templateData = {'device1' : devicests}
    return render_template('index.html', **templateData)

@app.route('/off')
def clear():
    devicests = "OFF"
    sense.clear()
    templateData = {'device1' : deviceSts}
    return render_template ('index.html', **templateData)

#Driver Code - Main Function :-
if _name_ == '_main_'
   app.run (host = [172.17.126.47]) # Port Number - 5000
                   ip config
```

→ 'static' Folder :- style.css -

```
body {
    background: blue;
    color : yellow;
}

.button {
   font : bold 15px Arial;
   text-decoration : none;
   background-color : #EEEEEE;
   color : #333333
   padding : 2px 6px 2px 6px;
   border-top : 1px solid #CCCCCC;
   border-right : 1px solid #333333;
   border-bottom : 1px solid #333333;
   border-left : 1px solid #CCCCCC;
}
```

→ 'templates' Folder :- index.html -

```
<!DOCTYPE html>
   <head>
       <title> Device Control </title>
       <link rel = "stylesheet" href = "/static/style.css" />
   </head>
   <body>
       <h1> Raspberry Pi Device Control </h1>
       <h2> Device status </h2>
       <h3> Device 1 ==> {{ device1 }} </h3>
```

* OUTPUTS :

→ index.html -

Raspberry Pi Device Control

Device Status

Device 1 ==> {{ device 1 }}

Commands

Device ctrl ==> Device 1 ON   Device 1 OFF

```
    <br>


    <h2> Commands </h2>
    <h3>
        Device Ctrl ==>
        <a href = "/d1/on" class= "button"> Device 1 ON </a>
        <a href = "/d1/off" class = "button"> Device 1 OFF </a>
    </h3>
  </body>

</html>
```

* **RESULT:**
Thus, implemented Web server and its interactions using Flask Framework on Raspberry Pi with SenseHAT. Thus, all the simulation results were verified successfully.