**Coding of the System**

Python

from flask import Flask, render\_template, jsonify

from datetime import datetime

import os

import time

import threading

import RPi.GPIO as GPIO

import dht11

import subprocess  # Import subprocess for executing shell commands

# Permissions and GPIO setup

os.system('sudo usermod -aG gpio kalpenmodi')

os.system('sudo chown root:gpio /dev/gpiomem')

os.system('sudo chmod 660 /dev/gpiomem')

path = os.path.abspath(\_\_file\_\_)

get\_dir\_path = os.path.dirname(path)

# GPIO PIN configurations

irsensor = 26  # GPIO26 for IR sensor

led = 21  # GPIO21 for LED

irstate = False

humidity = '00'

temperature = '00'

GPIO.setwarnings(False)

GPIO.setmode(GPIO.BCM)  # Use BCM numbering

GPIO.setup(irsensor, GPIO.IN, pull\_up\_down=GPIO.PUD\_DOWN)

GPIO.setup(led, GPIO.OUT)

# Create directory for captured images

if not os.path.exists(f"{get\_dir\_path}/captured\_images"):

    os.makedirs(f"{get\_dir\_path}/captured\_images")

# DHT11 thread for temperature and humidity

def dht11\_thread():

    global humidity, temperature

    while True:

        time.sleep(3)

        instance = dht11.DHT11(pin=14)

        result = instance.read()

        if result.is\_valid():

            temperature = ("%-3.1f C" % result.temperature)

            humidity = ("%-3.1f %%" % result.humidity)

        else:

            pass

# PIR sensor with LED and camera capture

def ir\_with\_led\_and\_camera():

    global irstate

    while True:

        time.sleep(0.5)

        if GPIO.input(irsensor) == 1:  # Motion detected

            GPIO.output(led, GPIO.HIGH)  # Turn LED ON

            irstate = True

            # Capture image using libcamera-jpeg

            timestamp = datetime.now().strftime("%Y-%m-%d\_%H-%M-%S")

            image\_path = f"{get\_dir\_path}/captured\_images/motion\_{timestamp}.jpg"

            subprocess.run(["libcamera-jpeg", "-o", image\_path])

            print(f"Motion detected! Image saved to {image\_path}")

        else:  # No motion detected

            GPIO.output(led, GPIO.LOW)  # Turn LED OFF

            irstate = False

# Threads for DHT11 and IR sensor

th2 = threading.Thread(target=dht11\_thread)

th2.start()

th1 = threading.Thread(target=ir\_with\_led\_and\_camera)

th1.start()

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

@app.route('/')

def home():

    return render\_template('index.html')

@app.route('/motion\_data')

def get\_motion\_data():

    global irstate, humidity, temperature

    if irstate:

        ir\_state = 'Detected'

        led\_status = 'ON'

    else:

        ir\_state = 'Not Detected'

        led\_status = 'OFF'

    motion\_data = {

        "Ir\_State": ir\_state,

        "led\_state": led\_status,

        "humidity": humidity,

        "temperature": temperature

    }

    return jsonify(motion\_data)

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(host='0.0.0.0', port=5003, debug=True)f

HTML

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Status</title>

<style>

body {

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

margin: 0;

font-family: 'Roboto', Arial, sans-serif;

background: linear-gradient(to bottom right, #000000, #000000, #000000);

}

.card {

background: #ffffff;

border-radius: 15px;

box-shadow: 0 8px 20px rgba(0, 0, 0, 0.2);

padding: 30px 20px;

width: 400px;

text-align: center;

transition: transform 0.3s ease, box-shadow 0.3s ease;

}

.card:hover {

transform: translateY(-5px);

box-shadow: 0 12px 30px rgba(0, 0, 0, 0.3);

}

.card h2 {

font-size: 26px;

margin-bottom: 25px;

color: #34495e;

}

.card p {

font-size: 18px;

margin: 10px 0;

display: flex;

justify-content: space-between;

color: #2c3e50;

}

.label {

font-weight: bold;

}

.value {

font-weight: normal;

color: #1abc9c;

transition: color 0.3s ease;

}

/\* Responsive Design \*/

@media (max-width: 400px) {

.card {

width: 90%;

padding: 20px;

}

}

</style>

<script>

async function fetchMotionData() {

try {

const response = await fetch('/motion\_data');

const data = await response.json();

// Update the UI directly without replacing content

updateValue('Ir\_State', data.Ir\_State);

updateValue('led\_state', data.led\_state);

updateValue('humidity', data.humidity);

updateValue('temperature', data.temperature);

} catch (error) {

console.error('Error fetching data:', error);

}

}

function updateValue(id, newValue) {

const element = document.getElementById(id);

if (element.textContent !== newValue) {

element.textContent = newValue;

// Add subtle color change animation

element.style.color = '#1abc9c'; // Temporary color

setTimeout(() => {

element.style.color = '#1abc9c'; // Transition back to green

}, 300);

}

}

// Fetch data every second

setInterval(fetchMotionData, 1000);

window.onload = fetchMotionData;

</script>

</head>

<body>

<div class="card">

<h2>Motion Status</h2>

<p><span class="label">Motion State:</span> <span class="value" id="Ir\_State">Loading...</span></p>

<p><span class="label">LED Status:</span> <span class="value" id="led\_state">Loading...</span></p>

<p><span class="label">Humidity:</span> <span class="value" id="humidity">Loading...</span></p>

<p><span class="label">Temperature:</span> <span class="value" id="temperature">Loading...</span></p>

</div>

</body>

</html>